



Universidad Nacional del Nordeste



Facultad de Ciencias Exactas y Naturales y Agrimensura
Carrera: Licenciatura en Sistemas de Información

Cátedra: Bases de Datos I
Año: 2022

Proyecto de Estudio: “Base de datos de Secuencias de Nucleótidos para la investigación genética”

Grupo N° 4

Integrantes:

Benítez Rinas, Juan Gabriel L.U.: 55527

Caballero, Federico David L.U.: 54396

Colman, Alexis Nicolas L.U.: 54341

Colunga, Raúl Rubén L.U.: 47336

Ponce, Leandro L.U.: 54236

ÍNDICE

CAPITULO I.....	5
Introducción	5
CAPITULO II: Marco conceptual o referencial	6
Antecedentes y conceptos de genética.....	6
Conceptos de ADN,GEN y Secuencia de nucleótidos	6
Ejemplo introductorio de una base genética.....	6
CAPITULO III: Metodología seguida.....	7
CAPITULO IV: Desarrollo del tema/resultados.....	8
Diagrama entidad relación.....	8
Relaciones	8
Relaciones de integridad referencial	10
Diccionario de datos	12
TEMAS TECNICOS.....	25
Procedimientos y funciones almacenadas.....	25
Disparados o Triggers.....	28
Vistas	30
Transacciones	33
Índices.....	38
Permisos	39
Backup	43
Recovery	43
CAPITULO V: Conclusión	51
CAPITULO VI: bibliografía	52

Índice de figuras

Figura 1. Concepto de comparación de secuencias	5
Figura 2. Representación esquemática de molécula de ADN	5
Figura 3. Esquema conceptual SEC1.....	7
Figura 4. Diagrama de Entidad Relación.....	8
Figura 5. comparación de secuencia.....	27
Figura 6. Esquema de backup propuesto para el proyecto.....	46

Índice de tablas

Tabla 1. Funcionamiento de triggers	30
Tabla 2. Funcionamiento de vistas.....	31
Tabla 3. Recoveries.....	43
Tabla 4. Esquema backup	45

CAPÍTULO I: INTRODUCCIÓN

Una de las características de la investigación genómica moderna es la generación de enormes cantidades de datos sin procesar de las secuencias de nucleótidos obtenidas. A medida que crece el volumen de los datos genómicos, sofisticadas metodologías computacionales son necesarias para administrar el diluvio de datos. Así, el primer reto de la era de la genómica es almacenar y manejar el enorme volumen de información a través de la creación y uso de bases de datos informáticos.

El posterior tratamiento de los datos allí almacenados permite el “descubrimiento de conocimiento”, referido a la identificación de conexiones entre piezas de información que no se conocían cuando se ingresó la información por primera vez. Este tratamiento se lleva a cabo empleando algoritmos y programas computacionales desarrollados para tal propósito. Solamente a modo de ejemplo: la comparación de n secuencias provenientes de especies distintas (pero equivalentes en cuanto a su localización, es decir, comparables) mediante algoritmos que emplean modelos probabilísticos, permite establecer la “distancia evolutiva” existente entre tales secuencias, y así generar árboles de parentesco inferidos según qué tan diferentes o similares sean tales secuencias.

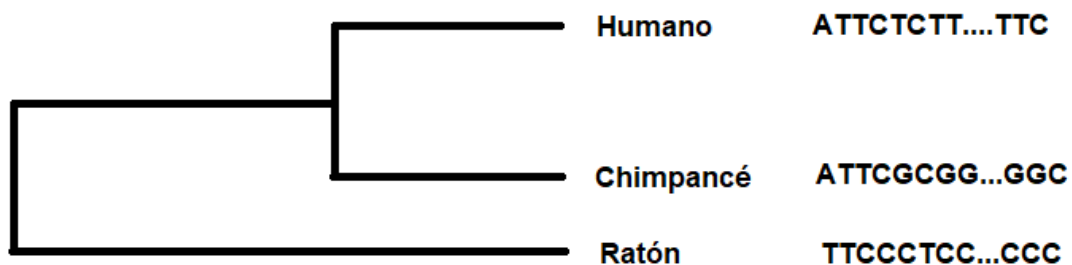


Figura 1. Concepto de comparación de secuencias.

Comparación, mediante algoritmos, de 3 cadenas de caracteres (de aprox. 16 mil caracteres cada una) que representan la **secuencia de nucleótidos** proveniente del genoma completo de mitocondria del humano, chimpancé y ratón. Generación de un “árbol de parentesco” mediante similitud de secuencias (adaptado de Posada, D. 2009).

En ese sentido, el principal dato a almacenar en la base de datos del presente proyecto corresponde a una **secuencia de nucleótidos**, que es una cadena de caracteres de longitud variable, conformada principalmente por los caracteres A, T, C y G. Tales caracteres representan los nucleótidos presentes en una muestra de ADN, cuya secuencia codifica información genética. Se debe, además, almacenar otros datos referidos a tal secuencia, como la especie del organismo muestreado (y sus datos taxonómicos: dominio, reino, filo, clase, etc.), si proviene del núcleo celular y el número de cromosoma al cual pertenece la secuencia, o si proviene de mitocondria o cloroplasto, el autor o autores de la secuencia, las publicaciones científicas en las cuales se utilizó una secuencia particular para realizar el estudio, etc.

El almacenamiento de estos registros de secuencias de nucleótidos junto a sus datos asociados, estructurados de manera apropiada, permitirá la posterior recuperación y tratamiento de tal información bajo distintos criterios de búsqueda, lo cual facilitará el seguimiento y progreso de las investigaciones de una determinada institución científica.

La correcta estructuración de los datos se refiere al correcto modelado del problema en cuestión y la reducción a tablas asociado, mediante el proceso de normalización, y

la implementación de los temas técnicos planteados para el presente trabajo, lo cual brindará todos los beneficios del modelo relacional.

CAPITULO II: MARCO CONCEPTUAL o REFERENCIAL

Antecedentes y conceptos de genética

Uno de los aspectos más utilizados en investigaciones de genética molecular y biología computacional es la comparación y alineamiento de secuencias de nucleótidos (Posada, 2009), es decir, comparaciones de meras cadenas de caracteres (que representan a las secuencias de nucleótidos) para descubrir genes y caracterizarlos dentro de tales secuencias, u otro tipo de estudios que comparan cadenas de caracteres (como aquel ejemplificado en la Figura 1).

Conceptos de ADN, Gen y Secuencia de nucleótidos

El ADN (ácido desoxirribonucleico) es el compuesto químico en el cual se encuentra codificada la información genética de todos los organismos. Dicha molécula está constituida por una sucesión de moléculas más pequeñas llamadas nucleótidos, como si fuesen las perlas de un collar de perlas. Las perlas serían los nucleótidos y el collar entero sería una molécula de ADN. Los nucleótidos que componen el ADN pueden ser de 4 tipos: **A**denina, **T**imina, **C**itosina y **G**uanina. La secuencia particular de tales nucleótidos dentro de una molécula de ADN es la que codifica la información genética, según el **código genético**.

Un **gen** es una porción de una molécula de ADN que va ser traducido (según el código genético) en una proteína. Podríamos entenderlo (abstrayéndonos de la complejidad química y biológica) como una porción de código informático con una función definida en el contexto de un programa mayor.

Molécula de ADN representada mediante su secuencia de nucleótidos

AATTGTGGGTCGTC TCGTGTA AAAATGCTGT...GGTTATTCGTATAGCTAT

Gen (en verde) representado mediante su secuencia de nucleótidos

Figura 2. Representación esquemática de molécula de ADN mediante su secuencia de nucleótidos. Dicha molécula alberga un gen (en verde). Dicho gen contiene la **información para producir una proteína específica**.

Ejemplo introductorio a la utilidad de una base de datos genética

Supongamos que en el marco de una investigación genética sencilla se ha obtenido una secuencia de nucleótidos de 120.000 caracteres perteneciente a la especie *Canis lupus*, y tal secuencia se obtuvo de *alguna* región del cromosoma 21 (dicha especie posee 78 cromosomas), y asignémosle arbitrariamente el nombre **SEC1** a tal secuencia. Supongamos además que los investigadores se encuentran “mapeando” el cromosoma 21 de tal especie, es decir, se encuentran secuenciando el cromosoma completo de a “pedacitos”, dado que éste es gigantesco, y SEC1 es uno de esos

“pedacitos”. Los investigadores desconocen completamente qué **genes** puede albergar SEC1, **si es que alberga genes** y si es que no se trata de una secuencia sin función aparente. Es aquí donde cobra utilidad el uso de herramientas informáticas para comparar secuencias de caracteres (es decir strings = varchar), dado que SEC1 se encuentra almacenada en nuestra base de datos bd_genetica en la tabla SecuenciaDeNucleotidos, y además disponemos de una tabla llamada Genes, que hipotéticamente contiene almacenada la información de miles o millones de genes **ya conocidos** de distintas especies, de los cuales se conoce su **secuencia** y su función. Por tanto, lo que podríamos hacer es comparar todas las secuencias de la tabla Genes con la secuencia de SEC1 en busca de alguna coincidencia, y en caso de encontrarse un ‘match’, volcar tal referencia en una tabla intermedia llamada Secuencia_Gen_Potencial, la cual va a indicar, por cada fila, que el Gen ‘X’ fue potencialmente encontrado en la Secuencia ‘SEC1’, el Gen ‘Z’ fue potencialmente encontrado en la Secuencia ‘SEC1’, etc. De esta manera, empleando un procedimiento almacenado podemos intentar encontrar genes ya conocidos y descriptos incluidos en SEC1, logrando así un primer intento de caracterizar la función biológica de ese trocito de ADN que los investigadores han secuenciado.

Secuencia de nucleótidos SEC1 (en azul)

ATTGCTGTCATAAACGGGGTATCGATTACCTAA....TTGGTGTGGCCACCA
 AACGGGGTATCG GGTGTGGCCA

Secuencias de los genes 'X' y 'Z' (en verde)

Figura 3. Esquema conceptual en el que se muestra la hipotética secuencia SEC1, junto a la secuencia de los genes 'X' y 'Z', que fueron hallados dentro de SEC1.

CAPÍTULO III: METODOLOGÍA

Para la realización del modelo de datos (primera entrega) del presente trabajo se realizaron intercambios por mensajería (*WhatsApp*) entre los integrantes del grupo, y una reunión a través de la aplicación Google Meet para la elección del tema principal del trabajo, y posteriormente se distribuyeron subtarefas para la elaboración del mismo. El Diagrama de Entidad Relación se elaboró usando la aplicación web *LucidChart*.

Para el abordaje y desarrollo de los temas técnicos (segunda entrega), en primer lugar, se subdividió la búsqueda e internalización de información respecto a los distintos temas (previo a aplicarlos al modelo de datos) entre los integrantes del grupo. Asimismo, se subdividió en tareas la escritura del código SQL para la creación de la base de datos (y carga de lote de datos) a partir del modelo, en SQL Management Studio. Posteriormente se realizaron reuniones vía *Google Meet* para decidir sobre la aplicación de los distintos temas técnicos al modelo de datos.

La búsqueda de información se realizó mediante consultas en páginas web, consulta de video tutoriales en la plataforma *YouTube* y también se consultaron libros citados

en la sección Bibliografía.

CAPITULO IV: DESARROLLO DEL TEMA/RESULTADOS

Diagrama Entidad Relación:

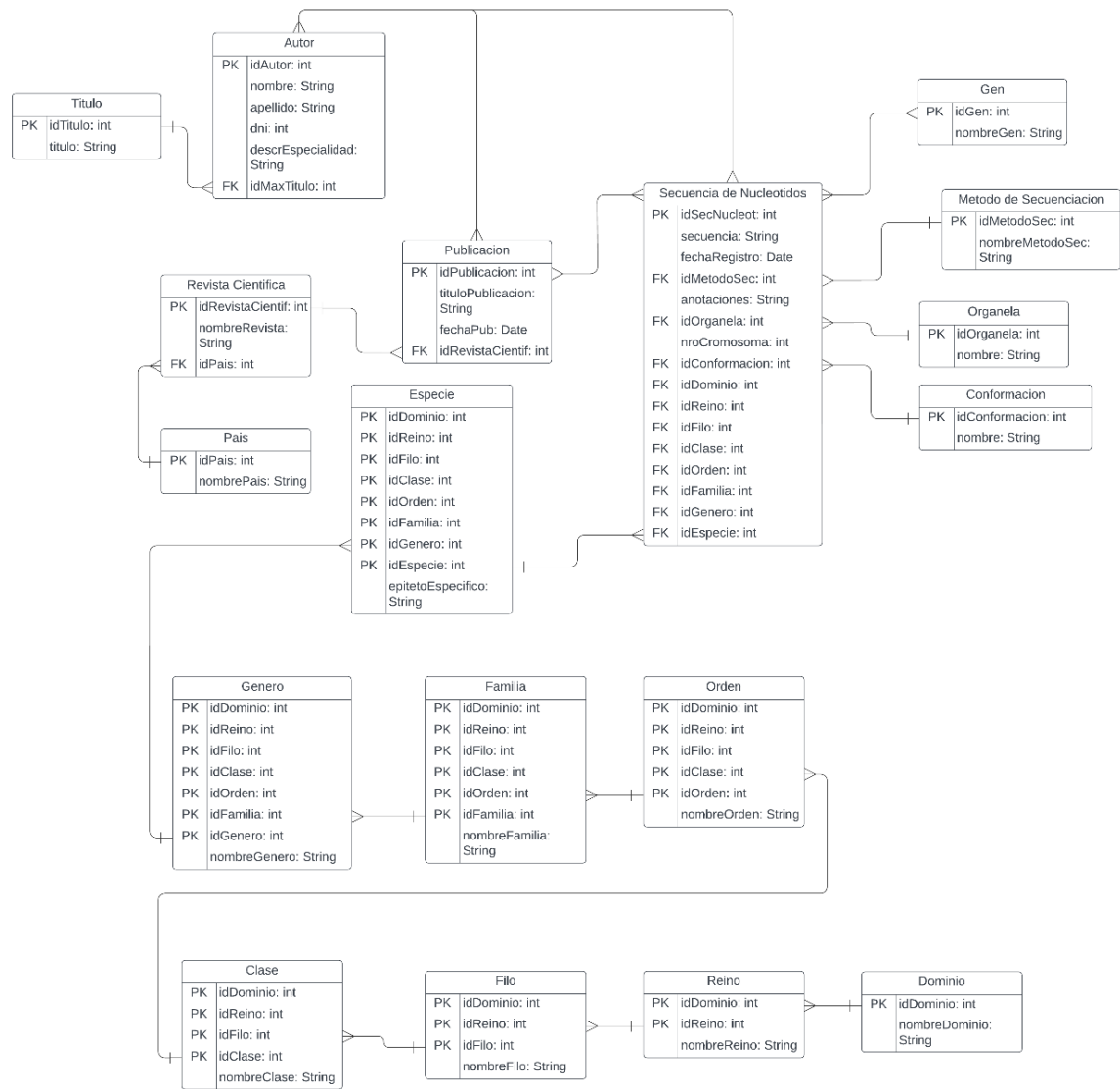


Figura 4. Diagrama de Entidad Relación.

Relaciones:

(A) Principal – Secuencia_de_Nucleotido

idSecNucleot	secuencia	fechaRegistro	idMetodoSec
PK - 1	2	3	FK - 4

anotaciones	idOrganela	nroCromosoma	idConformacion
5	FK - 6	7	FK - 8

idDominio	idReino	idFilo	idClase	idOrden
FK - 9	FK - 10	FK - 11	FK - 12	FK - 13

idFamilia	idGenero	idEspecie
FK - 14	FK - 15	FK - 16

(B) – Gen

idGen	nombreGen
PK - 1	2

(C) – Secuencia_Gen

idSecNucleot	idGen
PK - 1	PK - 2

(D) – Metodo_de_Secuenciacion

idMetodoSec	nombreMetodoSec
PK - 1	2

(E) – Organela

idOrganela	nombreOrganela
PK - 1	2

(F) – Conformación

idConformacion	nombreConformacion
PK - 1	2

(G) – Publicación

idPublicacion	tituloPublicacion	fechaPub	idRevistaCientif
PK - 1	2	3	FK - 4

(H) – Revista_Cientifica

idRevistaCientif	nombreRevista	idPais
PK - 1	2	FK - 3

(I) – Pais

idPais	nombrePais
PK - 1	2

(J) – Secuencia_Publicacion

idSecNucleot	idPublicacion
PK - 1	PK - 2

(K) – Autor

idAutor	nombre	apellido	dni	descrEspecialidad	idMaxTitulo
PK - 1	2	3	4	5	FK - 6

(L) – Titulo

idTitulo	titulo
PK - 1	2

(M)– Publicacion_Autor

idPublicacion	idAutor	numeroDeAutor
PK - 1	PK - 2	3

(N) Secuencia_Autor

idSecNucleot	idAutor
PK - 1	PK - 2

(O)– Especie

idDominio	idReino	idFilo	idClase	idOrden
PK - 1	PK - 2	PK - 3	PK - 4	PK – 5

idFamilia	idGenero	idEspecie	epitetoEspecifico
PK - 6	PK - 7	PK - 8	9

(P) – Genero

idDominio	idReino	idFilo	idClase	idOrden
PK - 1	PK - 2	PK - 3	PK - 4	PK – 5

idFamilia	idGenero	nombreGenero
PK - 6	PK - 7	PK - 8

(Q)– Familia

idDominio	idReino	idFilo	idClase	idOrden
PK - 1	PK - 2	PK - 3	PK - 4	PK – 5

idFamilia	nombreFamilia
PK - 6	7

(R)– Orden

idDominio	idReino	idFilo	idClase	idOrden	nombreOrden
PK - 1	PK - 2	PK - 3	PK - 4	PK - 5	6

(S)– Clase

idDominio	idReino	idFilo	idClase	nombreClase
PK - 1	PK - 2	PK - 3	PK - 4	5

(T)– Filo

idDominio	idReino	idFilo	nombreFilo
PK - 1	PK - 2	PK - 3	4

(U)– Reino

idDominio	idReino	nombreReino
PK - 1	PK - 2	3

(V)– Dominio

idDominio	nombreDominio
PK - 1	2

Restricciones de integridad referencial

A (4) → D (1)

A (6) → E (1)

A (8) → F (1)

A (9, 10, 11, 12, 13, 14, 15, 16) → O (1, 2, 3, 4, 5, 6, 7, 8)

A (9, 10, 11, 12, 13, 14, 15) → P (1, 2, 3, 4, 5, 6, 7)

A (9, 10, 11, 12, 13, 14) → Q (1, 2, 3, 4, 5, 6)

A (9, 10, 11, 12, 13) → R (1, 2, 3, 4, 5)

A (9, 10, 11, 12) → S (1, 2, 3, 4)

A (9, 10, 11) → T (1, 2, 3)

A (9, 10) → U (1, 2)

A (9) → V (1)

C (1) → A (1)

C (2) → B (1)

G (4) → H (1)

H (3) → I (1)

J (1) → A (1)

J (2) → G (1)

K (6) → L (1)

M (1) → G (1)

M (2) → K (1)

N (1) → A (1)

N (2) → K (1)

O (1, 2, 3, 4, 5, 6, 7) → P (1, 2, 3, 4, 5, 6, 7)

$O(1, 2, 3, 4, 5, 6) \rightarrow Q(1, 2, 3, 4, 5, 6)$
 $O(1, 2, 3, 4, 5) \rightarrow R(1, 2, 3, 4, 5)$
 $O(1, 2, 3, 4) \rightarrow S(1, 2, 3, 4)$
 $O(1, 2, 3) \rightarrow T(1, 2, 3)$
 $O(1, 2) \rightarrow U(1, 2)$
 $O(1) \rightarrow V(1)$
 $P(1, 2, 3, 4, 5, 6) \rightarrow Q(1, 2, 3, 4, 5, 6)$
 $P(1, 2, 3, 4, 5) \rightarrow R(1, 2, 3, 4, 5)$
 $P(1, 2, 3, 4) \rightarrow S(1, 2, 3, 4)$
 $P(1, 2, 3) \rightarrow T(1, 2, 3)$
 $P(1, 2) \rightarrow U(1, 2)$
 $P(1) \rightarrow V(1)$
 $Q(1, 2, 3, 4, 5) \rightarrow R(1, 2, 3, 4, 5)$
 $Q(1, 2, 3, 4) \rightarrow S(1, 2, 3, 4)$
 $Q(1, 2, 3) \rightarrow T(1, 2, 3)$
 $Q(1, 2) \rightarrow U(1, 2)$
 $Q(1) \rightarrow V(1)$
 $R(1, 2, 3, 4) \rightarrow S(1, 2, 3, 4)$
 $R(1, 2, 3) \rightarrow T(1, 2, 3)$
 $R(1, 2) \rightarrow U(1, 2)$
 $R(1) \rightarrow V(1)$
 $S(1, 2, 3) \rightarrow T(1, 2, 3)$
 $S(1, 2) \rightarrow U(1, 2)$
 $S(1) \rightarrow V(1)$
 $T(1, 2) \rightarrow U(1, 2)$
 $T(1) \rightarrow V(1)$
 $U(1) \rightarrow V(1)$

Diccionario de datos

(A) Secuencia_de_Nucleotido

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idSecNucleot	Identificación única de cada secuencia	int	Número natural	No aplica	NO
secuencia	La secuencia de nucleótidos en sí misma, representada por una cadena de caracteres	Varchar	Texto de longitud variable	Los caracteres permitidos son A, T, C, G, U, R, Y, K, M, S, W, B, D, H, V, N, X y -, siguiendo el formato	NO

				FASTA*	
fechaRegistro	Fecha en la que se registró la secuencia en la base de datos	Date	aaaa-mm-dd	No aplica	NO
idMetodoSec	Número de identificación del método de secuenciación empleado para obtener la secuencia de nucleótidos	int	1= Reacción en cadena de la polimerasa (PCR); 2= Método químico de Maxam y Gilbert; 3= Método enzimático de Sanger;	No aplica	NO
anotaciones	Anotaciones funcionales o de cualquier índole relacionadas a la secuencia	Varchar	Texto de longitud variable	No aplica	SI
idOrganela	Número de identificación de la organela de la cual se extrajo la secuencia	int	1=Núcleo; 2=Mitocondria; 3=Cloroplasto;	No aplica	NO
nroCromosoma	Número del cromosoma del cual se extrajo la secuencia, en caso de que ésta se haya extraído del núcleo de un organismo Eucarionte	int	Número natural	No aplica	SI
idConformacion	Conformación de la molécula de ADN de la cual se extrajo la secuencia	int	1=Circular; 2=Lineal;	No aplica	NO
idDominio	Nro. de id. del Dominio (correspondiente a la clasif. taxonómica) al	int	Número natural	No aplica	NO

	cual pertenece el organismo del cual se extrajo la secuencia				
idReino	Nro. de id. del Reino (correspondiente a la clasif. taxonómica) al cual pertenece el organismo del cual se extrajo la secuencia	int	Número natural	No aplica	NO
idFilo	Nro. de id. del Filo (correspondiente a la clasif. taxonómica) al cual pertenece el organismo del cual se extrajo la secuencia	int	Número natural	No aplica	NO
idClase	Nro. de id. De la Clase (correspondiente a la clasif. taxonómica) a la cual pertenece el organismo del cual se extrajo la secuencia	int	Número natural	No aplica	NO
idOrden	Nro. de id. del Orden (correspondiente a la clasif. taxonómica) al cual pertenece el organismo del cual se extrajo la secuencia	int	Número natural	No aplica	NO
idFamilia	Nro. de id. De la Familia (correspondiente a la clasif. taxonómica) a la cual pertenece el organismo del	int	Número natural	No aplica	NO

	cual se extrajo la secuencia				
idGenero	Nro. de id. del Género (correspondiente a la clasif. taxonómica) al cual pertenece el organismo del cual se extrajo la secuencia	int	Número natural	No aplica	NO
idEspecie	Nro. de id. de la especie de la cual se extrajo la secuencia	int	Número natural	No aplica	NO

*FASTA: El formato FASTA es un formato de fichero informático basado en texto, utilizado para representar secuencias de ácidos nucleicos, de péptido, y en el que los nucleótidos se representan usando códigos de una única letra.

(B) Gen

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idGen	Identificación única de cada gen	int	Número natural	No aplica	NO
nombreGen	Nombre del gen	Varchar	Texto de longitud variable	No aplica	NO

(C) Secuencia_Gen

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idSecNucleot	Identificación única de cada secuencia de la tabla Secuencia_de_Nucleotidos	int	Número natural	No aplica	NO
idGen	Identificación única de cada gen de la tabla Gen	int	Número natural	No aplica	NO

(D) Metodo_de_Secuenciacion

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idMetodoSec	Identificación única de cada método de secuenciación empleado para obtener secuencias	int	Número natural	No aplica	NO
nombreMetodoSec	Nombre del método de secuenciación	Varchar	Texto de longitud variable	No aplica	NO

(E) Organela

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idOrganela	Identificación única de cada organela	int	Número natural	No aplica	NO
nombre Organela	Nombre de la organela celular	Varchar	Texto de longitud variable	No aplica	NO

(F) Conformacion

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idConformacion	Identificación única de la conformación	int	Número natural	No aplica	NO
nombre Conformacion	Nombre de las posibles conforma-	Varchar	Texto de longitud variable	No aplica	NO

	ciones de las moléculas de ADN				
--	--------------------------------	--	--	--	--

(G) Publicacion

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idPublicacion	Identificación única de la publicación científica	int	Número natural	No aplica	NO
titulo Publicacion	Título de la publicación científica	Var-char	Texto de longitud variable	No aplica	NO
fechaPub	Fecha de publicación	Date	aaaa-mm-dd	No aplica	NO
idRevistaCientif	Identificación única de la revista científica en la cual se realizó la publicación	int	Número natural	No aplica	NO

(H) Revista_Cientifica

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idRevistaCientif	Identificación única de la revista científica	int	Número natural	No aplica	NO
nombreRevista	Nombre de la revista científica	Var-char	Texto de longitud variable	No aplica	NO
idPais	Identificación única del país al cual pertenece la revista científica	int	Número natural	No aplica	NO

(I) Pais

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idPais	Identificación única del país	int	Número natural	No aplica	NO
nombrePais	Nombre del país	varchar	Texto de longitud variable	No aplica	NO

(J) Secuencia_Publicacion

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idSecNucleot	Identificación única de la secuencia de nucleótidos utilizada en la publicación	int	Número natural	No aplica	NO
idPublicacion	Identificación única de la publicación que emplea la secuencia de nucleótidos	int	Número natural	No aplica	NO

(K) Autor

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idAutor	Identificación única de un autor de [publicación o, secuencia de nucleótidos]	int	Número natural	No aplica	NO
nombre	Nombre del autor	Varchar	Texto de longitud variable	No aplica	NO
apellido	Apellido del autor	Varchar	Texto de longitud variable	No aplica	NO
dni	Número de	int	Número	Debe contener	NO

	documento nacional de identidad del autor		natural	exactamente 7 u 8 dígitos	
descr Especialidad	Descripción breve de la especialidad científica en la cual se desempeña el autor	Varchar	Texto de longitud variable	No aplica	SI
idMaxTitulo	Identificación única del máximo título alcanzado por el autor	int	Número natural	No aplica	NO

(L) Titulo

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idTitulo	Identificación única del título académico de pregrado, grado o posgrado	int	Número natural	No aplica	NO
titulo	Denominación del título académico de pregrado, grado o posgrado	Varchar	Texto de longitud variable	No aplica	NO

(M) Publicacion_Autor

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idPublicacion	Identificación única de la publicación en la cual participó el autor	int	Número natural	No aplica	NO

idAutor	Identificación única del autor que participó de la publicación	int	Número natural	No aplica	NO
numeroDeAutor	Número que representa la posición que ocupa el autor en la lista de autores de la publicación, pudiendo ser 1: autor principal, o >1: autor secundario.	int	Número natural	No aplica	NO

(N) Secuencia_Autor

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idSecNucleot	Identificación única de la secuencia de nucleótidos de la cual el autor es responsable	int	Número natural	No aplica	NO
idAutor	Identificación única del autor de la secuencia de nucleótidos	int	Número natural	No aplica	NO

(O) Especie

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al	int	Número natural	No aplica	NO

	cual pertenece la especie				
idReino	Identificación única del Reino al cual pertenece la especie	int	Número natural	No aplica	NO
idFilo	Identificación única del Filo al cual pertenece la especie	int	Número natural	No aplica	NO
idClase	Identificación única la Clase al cual pertenece la especie	int	Número natural	No aplica	NO
idOrden	Identificación única del Orden al cual pertenece la especie	int	Número natural	No aplica	NO
idFamilia	Identificación única de la Familia a la cual pertenece la especie	int	Número natural	No aplica	NO
idGenero	Identificación única del Género al cual pertenece la especie	int	Número natural	No aplica	NO
idEspecie	Identificación única de la especie	int	Número natural	No aplica	NO
epiteto Especifico	Corresponde al segundo término del nombre científico de una especie. El nombre científico se	Varchar	Texto de longitud variable	No aplica	NO

	compone del nombre del Género + epíteto específico				
--	--	--	--	--	--

(P) Genero

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al cual pertenece el Género	int	Número natural	No aplica	NO
idReino	Identificación única del Reino al cual pertenece el Género	int	Número natural	No aplica	NO
idFilo	Identificación única del Filo al cual pertenece el Género	int	Número natural	No aplica	NO
idClase	Identificación única la Clase al cual pertenece el Género	int	Número natural	No aplica	NO
idOrden	Identificación única del Orden al cual pertenece el Género	int	Número natural	No aplica	NO
idFamilia	Identificación única de la Familia a la cual pertenece el Género	int	Número natural	No aplica	NO
idGenero	Identificación única del Género	int	Número natural	No aplica	NO

nombreGenero	Nombre del Género	Var-char	Texto de longitud variable	No aplica	NO
--------------	-------------------	----------	----------------------------	-----------	----

(Q) Familia

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al cual pertenece la Familia	int	Número natural	No aplica	NO
idReino	Identificación única del Reino al cual pertenece la Familia	int	Número natural	No aplica	NO
idFilo	Identificación única del Filo al cual pertenece la Familia	int	Número natural	No aplica	NO
idClase	Identificación única la Clase al cual pertenece la Familia	int	Número natural	No aplica	NO
idOrden	Identificación única del Orden al cual pertenece la Familia	int	Número natural	No aplica	NO
idFamilia	Identificación única de la Familia	int	Número natural	No aplica	NO
nombreFamilia	Nombre de la Familia	Var-char	Texto de longitud variable	No aplica	NO

(R) Orden

Atributo	Descripción	Tipo de	Dominio	Especificación	Acepta NULL
----------	-------------	---------	---------	----------------	-------------

		dato			
idDominio	Identificación única del Dominio al cual pertenece el Orden	int	Número natural	No aplica	NO
idReino	Identificación única del Reino al cual pertenece el Orden	int	Número natural	No aplica	NO
idFilo	Identificación única del Filo al cual pertenece el Orden	int	Número natural	No aplica	NO
idClase	Identificación única la Clase al cual pertenece el Orden	int	Número natural	No aplica	NO
idOrden	Identificación única del Orden	int	Número natural	No aplica	NO
nombreOrden	Nombre del Orden	Var-char	Texto de longitud variable	No aplica	NO

(S) Clase

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al cual pertenece la Clase	int	Número natural	No aplica	NO
idReino	Identificación única del Reino al cual pertenece la Clase	int	Número natural	No aplica	NO
idFilo	Identificación única del	int	Número natural	No aplica	NO

	Filo al cual pertenece la Clase				
idClase	Identificación única de la Clase	int	Número natural	No aplica	NO
nombreClase	Nombre de la Clase	Var-char	Texto de longitud variable	No aplica	NO

(T) Filo

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al cual pertenece el Filo	int	Número natural	No aplica	NO
idReino	Identificación única del Reino al cual pertenece el Filo	int	Número natural	No aplica	NO
idFilo	Identificación única del Filo	int	Número natural	No aplica	NO
nombreFilo	nombre del Filo	Var-char	Texto de longitud variable	No aplica	NO

(U) Reino

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio al cual pertenece el Reino	int	Número natural	No aplica	NO
idReino	Identificación única del Reino	int	Número natural	No aplica	NO

nombreReino	Nombre del Reino	Varchar	Texto de longitud variable	No aplica	NO
-------------	------------------	---------	----------------------------	-----------	----

(V) Dominio

Atributo	Descripción	Tipo de dato	Dominio	Especificación	Acepta NULL
idDominio	Identificación única del Dominio de la taxonomía biológica	int	Número natural	No aplica	NO
nombreDominio	Nombre del dominio	Varchar	Texto de longitud variable	No aplica	NO

TEMAS TÉCNICOS

PROCEDIMIENTOS y FUNCIONES ALMACENADAS

A continuación, se ilustra lo que ha sido descrito en el CAPITULO II, implementado de manera sencilla con una función y dos procedimientos almacenados:

```
CREATE FUNCTION F_buscarGenPotencial (@pidGen int) RETURNS TABLE
AS
RETURN
    (SELECT idSecNucleot
     FROM SecuenciaDeNucleotidos
     WHERE secuencia like concat('%',(SELECT secuencia FROM Gen WHERE idGen =
@pidGen) , '%'));
```

La figura x muestra una función que efectúa una consulta sobre la tabla SecuenciaDeNucleótidos y la filtra con el operador 'like' para obtener todos los registros de tal tabla en cuya secuencia se halle albergada la secuencia del Gen ingresado como parámetro. La función devuelve una tabla de una sola columna. Cada registro de tal columna corresponde al idSecNucleot (PK de la tabla SecuenciaDeNucleotidos) para la cual se encontró la secuencia del Gen ingresado como parámetro.

Luego empleamos esta función dentro de un procedimiento almacenado, para iterar las comparaciones:

```

CREATE PROCEDURE SP_almacenarGenPotencial
@pidGen int
AS
declare @temp table(idSecNucleot int);
insert into @temp(idSecNucleot) select idSecNucleot from
F_buscarGenPotencial(@pidGen); --Utiliza la funcion previa
declare @count int = (select count(*) from @temp);

while @count > 0
BEGIN
    declare @idSecNucleot int = (select top(1) idSecNucleot from @temp order
by idSecNucleot)
    INSERT INTO Secuencia_Gen_Potencial(idGen, idSecNucleot)
VALUES(@pidGen,@idSecNucleot);
    delete @temp where idSecNucleot = @idSecNucleot;
    set @count = (select count(*) from @temp);
END

```

El procedimiento de la figura x utiliza la función previa y almacena la tabla obtenida desde tal función en una variable temporal @temp, que será utilizada para iterar dentro de un bucle while. El bucle simplemente itera sobre cada registro obtenido con la función almacenada, y por cada uno, almacena la referencia de idGen e idSecNucleot en la tabla intermedia Secuencia_Gen_Potencial. Con esta función y procedimiento se logra descubrir si la secuencia del registro de la tabla Gen ingresado como parámetro existe dentro de la secuencia de algún/os registro/s de la tabla SecuenciaDeNucleótidos, y de existir, se almacena esa información en otra tabla para su posterior análisis.

También podría haberse creado una función que tome como entrada el idSecNucleot y compare esa secuencia con la secuencia de todos los registros de la tabla Gen, es decir, como se observa en la figura x, podríamos ingresar un registro de la tabla SecuenciaDeNucleótidos como parámetro para que compare con todos los registros de la tabla Gen, o vice versa, según la necesidad del investigador.

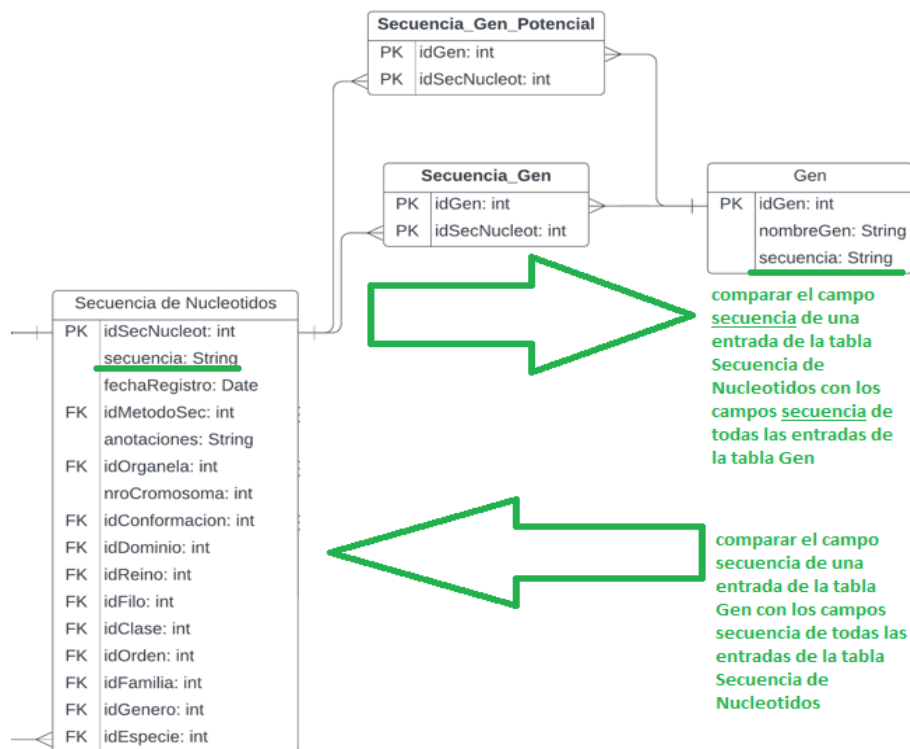


Figura 5: comparar 1-a-muchos en uno u otro sentido, según la necesidad del investigador.

Podemos llevar este análisis más allá y establecer un procedimiento almacenado que se ejecute automáticamente en determinado horario, de manera rutinaria, y busque coincidencias, pero esta vez comparando las secuencias de todos los Genes con las secuencias de todos los registros de la tabla **SecuenciaDeNucleotidos**, y en caso de hallar coincidencias, nuevamente, que almacene las referencias en la tabla intermedia **Secuencia_Gen_Potencial**. De esta manera se automatiza la búsqueda de genes dentro de los registros de la tabla **SecuenciasDeNucleotidos**, que son **secuencias desconocidas** que van a ir siendo cargadas continuamente dentro de la base de datos.

```
CREATE PROCEDURE SP_escanearGenes
AS
declare @temp table(idGen int);
insert into @temp(idGen) select idGen from Gen;
declare @count int = (select count(*) from @temp);
while @count > 0
BEGIN
    declare @idGen int = (select top(1) idGen from @temp order by idGen)
    EXEC SP_buscarGenPotencial_test @idGen
    delete @temp where idGen = @idGen;
```

```
set @count = (select count(*) from @temp);
```

END

En el caso del procedimiento SP_escanearGenes, se utiliza una variable temporal que almacena la tabla Genes, para iterar sobre dicha tabla en un bucle while. En el bucle while, se ejecuta el procedimiento almacenado SP_buscarGenPotencial_test previamente descrito, una vez por cada registro de la tabla Genes. Con esto se logra comparar todos los registros de la tabla Gen con todos los registros de la tabla SecuenciaDeNucleotidos, almacenando las coincidencias halladas en la tabla Secuencia_Gen_Potencial, es decir, un escaneo completo de la tabla principal de la base de datos.

Estos procedimientos y funciones podrían refinarse, por ejemplo, creando subcategorías en la tabla Genes para evitar comparaciones innecesarias, o refinando el algoritmo para que no busque solamente coincidencias exactas, sino alineamientos aproximados, dado que un mismo gen (por ejemplo, el gen de la hemoglobina) no va a tener exactamente la misma secuencia en diversas especies, aunque si puede ser muy similar en especies emparentadas. Tales refinamientos escapan al análisis introductorio del presente trabajo.

Asimismo, se puede mejorar la velocidad de ejecución de tales procedimientos empleando un **índice fulltext** sobre el campo secuencia de las tablas SecuenciaDeNucleotidos y Gen, y utilizando las funciones **contains** o **freetext** en lugar del operador 'like'.

DISPARADORES o TRIGGERS

Los triggers pueden ser utilizados para forzar las reglas de negocio en una base de datos. En nuestro caso podemos utilizar un trigger para la tabla Gen para inserción o modificación, que verifique que el varchar insertado/modificado en el campo secuencia presente el formato correcto, es decir: además de obviamente ser del tipo de dato correcto varchar, debe presentar únicamente los caracteres 'A', 'G', 'C' y/o 'T'. Un trigger idéntico puede utilizarse para la tabla SecuenciaDeNucleotidos, pero omitimos mostrarlo por ser redundante.

Para implementar dicho trigger, hemos codificado previamente una función almacenada, la cual toma como parámetro un varchar, y devuelve un número que es igual a 0 en caso de que el varchar tenga el formato correcto, o un valor mayor que 0 en caso de que no tenga el formato correcto.

```
CREATE FUNCTION F_secError(@secuencia varchar(Max)) RETURNS int
```

AS

BEGIN

```
RETURN LEN(TRIM(REPLACE(REPLACE(REPLACE(REPLACE(@secuencia, 'A', ''), 'T',  
''), 'C', ''), 'G', '')))
```

END

La función reemplaza los caracteres A, T, C y G por ausencia de carácter (elimina el carácter), y luego devuelve la longitud del varchar resultante, con lo cual, si existen caracteres distintos a los antes mencionados, devolverá un valor mayor que 0.

Comprobamos la ejecución de la función:

```
SELECT      dbo.F_secError('ATTATCGGTGGTAC')      as      'Sin      error',
dbo.F_secError('ATTATCGJGTGGTAQC') as 'Con error';
```

Results			Messages	
	Sin error	Con error		
1	0	2		

Luego, usamos esa función dentro del trigger:

```
CREATE TRIGGER TR_check_formato_secuencia
ON Gen
INSTEAD OF INSERT
AS
IF (SELECT dbo.F_secError(sequencia) FROM inserted) = 0
    BEGIN
        declare @nombre varchar(50) = (SELECT nombre FROM inserted)
        declare @sequencia varchar(Max) = REPLACE((SELECT sequencia FROM
inserted), ' ', '')
        INSERT INTO Gen(nombre, sequencia) VALUES(@nombre, @sequencia)
        print('Gen insertado');
    END
ELSE
    print('Formato de secuencia del Gen INCORRECTO. No fue posible insertar el Gen en
la base de datos. Revise la secuencia, dado que existen caracteres
incompatibles.');
```

Este trigger, además de impedir la inserción en caso de que haya caracteres incorrectos en la secuencia, también elimina los espacios en blanco que puedan haber en una secuencia correcta, antes de insertar.

Por otra parte, se ilustra nuevamente la posibilidad de usar una función almacenada dentro de un trigger. Si bien esto podría haberse hecho sin crear una función, pero es una ventaja poder modularizar el código, además de aportar a la legibilidad del mismo.

Ahora, comprobamos que el trigger funcione correctamente, ejecutamos:

```
INSERT INTO Gen(nombre, secuencia) VALUES('GenCorrecto', 'TTGTTATTGTGAACA');
INSERT INTO Gen(nombre, secuencia) VALUES('GenCorrecto2', 'TTGGGTCA CCACGATCA');
INSERT INTO Gen(nombre, secuencia) VALUES('GenError1', 'TTGG.GTCACCACGATCA');
INSERT INTO Gen(nombre, secuencia) VALUES('GenError3', 'TTGGGT CA,CCA 6CGATCA');
```

Obtenemos:

```
Messages
(1 row affected)
Gen insertado

(1 row affected)

(1 row affected)
Gen insertado

(1 row affected)
Formato de secuencia del Gen INCORRECTO. No fue posible insertar el Gen en la base de datos. Revise la secuencia, dado que existen c

(1 row affected)
Formato de secuencia del Gen INCORRECTO. No fue posible insertar el Gen en la base de datos. Revise la secuencia, dado que existen c
```

	idGen	nombre	secuencia
▶	6	ACTA1	AAATTATAAAAA...
	7	MYO7A	TTTCTCATATC...
	8	ALK1	TTTATATATTAG...
	9	FOXP2	TTTAACCCAC...
	10	microRNA 17	GTTACGTCGTA...
	11	TESTEO	TTGACA
	12	TESTEO2	TATATATATAT...
	13	TESTEO3	TTAAATTTCGCG...
	23	GenCorrecto	TTGTTATTGTGA...
	24	GenCorrecto2	TTGGGTCA CC...
*	NULL	NULL	NULL

Tabla 1. Funcionamiento de Triggers

Los triggers también pueden sernos de utilidad para auditar los cambios realizados en nuestra base de datos, almacenando el usuario, fecha/hora y tipo de cambio realizados sobre una tabla, en otra tabla creada para tal fin.

////////////////////////////////////

VISTAS

Podemos implementar algunas vistas para agregar tablas, lo cual nos permite agilizar la escritura de consultas, e incluso la construcción de funciones, procedimientos o triggers empleando esas vistas. Por ejemplo, para visualizar los datos taxonómicos de todas las especies, podemos implementar la siguiente vista:

```

CREATE VIEW vw_especies_taxon
AS

SELECT dom.nombre 'Dominio', r.nombre 'Reino', fil.nombre 'Filo', c.nombre
'Clase',

o.nombre 'Orden', f.nombre 'Familia', g.nombre 'Género', sp.epitetoEspecifico
'Epíteto específico',

sp.idEspecie, sp.idGenero, sp.idFamilia, sp.idOrden, sp.idClase, sp.idFilo,
sp.idReino, sp.idDominio

FROM Especie sp INNER JOIN Genero g ON (sp.idGenero = g.idGenero)

INNER JOIN Familia f ON (g.idFamilia = f.idFamilia)

INNER JOIN Orden o ON (f.idOrden = o.idOrden)

INNER JOIN Clase c ON (c.idClase = o.idClase)

INNER JOIN Filo fil ON (c.idFilo = fil.idFilo)

INNER JOIN Reino r ON (fil.idReino = r.idReino)

INNER JOIN Dominio dom ON (r.idDominio = dom.idDominio)

```

Consultamos la vista:

```
SELECT * FROM vw_especies_taxon;
```

Y obtenemos:

	Dominio	Reino	Filo	Clase	Orden	Familia	Género	Epíteto específico	idEspecie	idGenero	idFamilia	idOrden	idClase	idFilo	idReino
1	Eukarya	Animalia	Chordata	Mammalia	Primates	Atelidae	Alouatta	caraya	2	2	3	1	1	1	1
2	Eukarya	Animalia	Chordata	Mammalia	Primates	Atelidae	Oreonax	flavicauda	3	3	3	1	1	1	1
3	Eukarya	Animalia	Chordata	Mammalia	Camivora	Canidae	Canis	familiaris	4	5	6	2	1	1	1
4	Eukarya	Animalia	Chordata	Mammalia	Camivora	Canidae	Canis	lupus	5	5	6	2	1	1	1
5	Eukarya	Animalia	Chordata	Mammalia	Primates	Hominidae	Homo	sapiens	1	1	1	1	1	1	1
6	Eukarya	Animalia	Arthropoda	Insecta	Lepidoptera	Nymphalidae	Archaeoprepona	demophoon	6	6	7	6	4	2	1

Tabla 2. Funcionamiento de vistas

De esta manera filtramos algunos datos redundantes que se van repitiendo entre estas tablas y mostramos la jerarquía de clasificación de manera ordenada. Esta vista puede luego utilizarse como base para otros objetos.

Lo mismo hacemos con las demás tablas, para disponer de dos vistas estándar para acceder a la información de las tablas:

```

CREATE VIEW vw_vistaSecNucleotidos
AS

SELECT sec.idSecNucleot, sec.secuencia, sec.fechaRegistro,

```



```

ms.nombre 'metodo_secuenciacion', org.nombre 'organela', conf.nombre
'conformacion', nroCromosoma, concat(sp.Género, ' ', sp.[Epíteto específico]) as
'Especie',

sec.anotaciones

FROM SecuenciaDeNucleotidos as sec

INNER JOIN MetodoDeSecuenciacion as ms ON (sec.idMetodoSec = ms.idMetodoSec)

INNER JOIN Organela as org ON (org.idOrganela = sec.idOrganela)

INNER JOIN Conformacion as conf ON (conf.idConformacion = sec.idConformacion)

INNER JOIN vw_especies_taxon as sp ON (

    sec.idDominio = sp.idDominio AND sec.idReino = sp.idReino AND sec.idFilo =
    sp.idFilo AND sec.idClase = sp.idClase

    AND sec.idOrden = sp.idOrden AND sec.idFamilia = sp.idFamilia AND
    sec.idGenero = sp.idGenero

    AND sec.idEspecie = sp.idEspecie

)

```

Vemos que es posible utilizar la vista vw_especies_taxon para construir la vista vw_vistaSecNucleotidos

Al ejecutar,

```
SELECT * FROM vw_vistaSecNucleotidos
```

Obtenemos:

Results Messages Live Query Statistics									
	idSecNu...	secuencia	fechaRegistro	metodo_secuenciacion	organela	conformacion	nroCromosoma	Especie	anotaciones
1	1	ATCGTGTGGCCCTTTTAAACG...	2015-03-01	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
2	2	ATCGTATTGGCCGAGGAGACAG...	2018-03-01	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
3	3	ATCGTATTGGCCGAGGAGACAG...	2012-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
4	4	ATCGTATTGGCCGAGGTTGACA...	2012-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
5	5	ATCGTATTGGCCGAGGAGACAG...	2012-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
6	6	ATCGTATTGGCCGAGGAGACAG...	2012-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
7	7	ATTATATTACCGAACTGTATATA...	2009-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
8	8	TATATATATATATATATATAT...	2009-04-21	Método Químico de Maxam y Gilbert	Núcleo	Lineal	2	Homo sapiens	NULL
9	9	AAGGGGGGGG	2001-05-13	Método Enzimático de Sanger	Mitocondria	Lineal	3	Canis familiaris	asd
10	11	TGTGTGGTGTGTG	2001-05-13	Método Enzimático de Sanger	Mitocondria	Lineal	3	Canis familiaris	asd

Activar Windows

Query executed successfully. DESKTOP-4FGE28G\SQLEXPRESS ... DESKTOP-4FGE28G\raulr... | bd_genetica | 00:00:01 | 10 rows

Esta vista nos facilita en gran medida filtrar las consultas que efectuamos sobre la tabla principal de la base de datos, por ejemplo, ahora podemos realizar la siguiente consulta sobre la vista:

```

SELECT idSecNucleot, secuencia FROM vw_vistaSecNucleotidos
WHERE Especie = 'Canis familiaris'

```

Obtenemos:

	idSecNucleot	secuencia
1	9	AAGGGGGGGG
2	11	TGTGTGGTGTG

O podríamos efectuar una consulta que, sin emplear la vista, sería muy tediosa de escribir, por ejemplo:

```
SELECT idSecNucleot, secuencia, Especie FROM vw_vistaSecNucleotidos
WHERE Especie like 'Rhinella %' AND nroCromosoma = 2 AND fechaRegistro BETWEEN
'20150523' AND '20200523'
```

Obtenemos:

	idSecNucleot	secuencia	Especie
1	13	TGTCCGCTTGGTATATTTTATTACGGTGTG	Rhinella marina
2	15	AGAGGTTAGATTGACCG	Rhinella alata

De esta manera, vemos que se simplifica en gran medida la escritura de la consulta al realizarla sobre la vista, a lo que sería realizar la misma consulta directamente sobre las tablas.

Las vistas también pueden utilizarse para ocultar columnas con datos sensibles, lo cual no resulta de mayor aplicabilidad en nuestro modelo, aunque podemos efectuarlo sobre la tabla Autor de la siguiente manera:

```
CREATE VIEW vw_autor
AS
SELECT idAutor, nombre, apellido
FROM Autor
```

Luego, podemos restringir para que ciertos usuarios puedan acceder solamente a esta vista y no a la tabla base y de esa forma ocultamos un campo que puede ser considerado sensible (dni) y otros campos que podrían no ser relevantes (descrEspecialidad e idMaxTitulo).

TRANSACCIONES

Para ejemplificar el uso de las transacciones para asegurar la correcta inserción en múltiples tablas, vamos a construir el siguiente procedimiento almacenado, integrando varios de los temas técnicos hasta aquí desarrollados.

Primero, desarrollamos las siguientes dos funciones, que permiten retornar una tabla (de una sola fila, dada la unicidad del campo ingresado para la búsqueda) según el nombre científico de la especie en el primer caso, y según DNI en el segundo:

```
CREATE FUNCTION buscarEspecie (@genero varchar(50), @epEsp varchar(50)) RETURNS
TABLE
AS

RETURN

SELECT * FROM vw_especies_taxon

WHERE (Género = @genero) AND ([Epíteto específico] = @epEsp)
```

```
CREATE FUNCTION buscarAutor (@dniAutor int) RETURNS TABLE
AS

RETURN

SELECT * FROM Autor

WHERE (dni = @dniAutor)
```

Luego, el procedimiento almacenado:

```
CREATE PROCEDURE SP_insertarSecuencia7 (@secuencia varchar(Max), @fechaRegistro
Date, @metodoSec varchar(50),

@anotaciones varchar(300), @organela varchar(30), @cromosoma int, @conformacion
varchar(30), @familia varchar(50), @genero varchar(50), @epEsp varchar(50),
@nombreAutor varchar(50), @apeAutor varchar(50), @dniAutor int)
AS

declare @idConformacion int = (SELECT idConformacion FROM Conformacion
WHERE nombre = @conformacion);

declare @idOrganela int = (SELECT idOrganela FROM Organela WHERE nombre =
@organela);

declare @idMetodoSec int = (SELECT idMetodoSec FROM MetodoDeSecuenciacion
WHERE nombre like concat('%',@metodoSec,'%'));

declare @idEspecie int;

declare @idGenero int;

declare @idFamilia int;

declare @idOrden int;

declare @idClase int;

declare @idFilo int;

declare @idReino int;
```

```

declare @idDominio int;

IF (SELECT COUNT(*) FROM buscarEspecie(@genero, @epEsp)) > 0 AND (SELECT COUNT(*)
FROM dbo.buscarAutor(@dniAutor)) > 0

    --el autor y especie ingresados ya existen en la bd, por lo cual no se
insertarán

    BEGIN

        SET @idEspecie = (SELECT idEspecie FROM buscarEspecie(@genero, @epEsp));
        SET @idGenero = (SELECT idGenero FROM buscarEspecie(@genero, @epEsp));
        SET @idFamilia = (SELECT idFamilia FROM buscarEspecie(@genero, @epEsp));
        SET @idOrden = (SELECT idOrden FROM buscarEspecie(@genero, @epEsp));
        SET @idClase = (SELECT idClase FROM buscarEspecie(@genero, @epEsp));
        SET @idFilo = (SELECT idFilo FROM buscarEspecie(@genero, @epEsp));
        SET @idReino = (SELECT idReino FROM buscarEspecie(@genero, @epEsp));
        SET @idDominio = (SELECT idDominio FROM buscarEspecie(@genero, @epEsp));

        INSERT INTO SecuenciaDeNucleotidos(secuencia, fechaRegistro, idMetodoSec,
        anotaciones, idOrganela, nroCromosoma, idConformacion, idEspecie,
        idGenero, idFamilia, idOrden, idClase, idFilo,

        idReino, idDominio)

        VALUES(@secuencia, @fechaRegistro, @idMetodoSec, @anotaciones,
        @idOrganela, @cromosoma, @idConformacion,

        @idEspecie, @idGenero, @idFamilia, @idOrden, @idClase, @idFilo, @idReino,
        @idDominio);

    END

ELSE

    BEGIN

        IF ((SELECT COUNT(*) FROM buscarEspecie(@genero, @epEsp)) > 0)

            BEGIN

                --Se encontró la especie, pero no el autor: Se inserta en la
tabla Autor y en la tabla SecuenciaDeNucleotidos

                --omitimos el código de esta parte por ser redundante

                PRINT('código omitido.');
```

```

--Se encontro el autor, pero no la especie: se inserta
en las tablas de especie (Especie, Género, Familia,etc y en
-- la tabla SecuenciaDeNucleótidos)
--omitimos el código de esta parte por ser redundante
PRINT('código omitido.');
```

END

ELSE

BEGIN

```

SET @idFamilia = (SELECT idFamilia FROM Familia WHERE
nombre = @familia);
SET @idOrden = (SELECT idOrden FROM Familia WHERE
nombre = @familia);
SET @idClase = (SELECT idClase FROM Familia WHERE
nombre = @familia);
SET @idFilo = (SELECT TOP(1) idFilo FROM Familia WHERE
nombre = @familia);
SET @idReino = (SELECT TOP(1) idReino FROM Familia
WHERE nombre = @familia);
SET @idDominio = (SELECT TOP(1) idDominio FROM Familia
WHERE nombre = @familia);
declare @error int = 0;

BEGIN TRAN

INSERT INTO Autor(nombre, apellido, dni)
VALUES(@nombreAutor, @apeAutor, @dniAutor);
SET @error = @error + @@ERROR;

INSERT INTO Genero(idFamilia, idOrden, idClase,
idFilo, idReino, idDominio, nombre)
VALUES(@idFamilia, @idOrden, @idClase, @idFilo,
@idReino, @idDominio, @genero);
SET @error = @error + @@ERROR;
SET @idGenero = (SELECT idGenero FROM Genero
WHERE nombre = @genero);

INSERT INTO Especie(idGenero, idFamilia,
idOrden, idClase, idFilo, idReino, idDominio, epitetoEspecifico)

```

```

VALUES(@idGenero, @idFamilia, @idOrden,
@idClase, @idFilo, @idReino, @idDominio, @epEsp);

SET @idEspecie = (SELECT idEspecie FROM Especie
WHERE epitetoEspecifico = @epEsp);

SET @error = @error + @@ERROR;

INSERT INTO SecuenciaDeNucleotidos(secuencia,
fechaRegistro, idMetodoSec,
anotaciones, idOrganela, nroCromosoma,
idConformacion, idEspecie, idGenero, idFamilia, idOrden, idClase, idFilo,
idReino, idDominio)
VALUES(@secuencia, @fechaRegistro,
@idMetodoSec, @anotaciones, @idOrganela, @cromosoma, @idConformacion,
@idEspecie, @idGenero, @idFamilia, @idOrden,
@idClase, @idFilo, @idReino, @idDominio);

SET @error = @error + @@ERROR;

IF(@error = 0)
    commit;
ELSE
    PRINT('La transacción no pudo
realizarse.');
```

rollback tran;

END

END

Explicación del procedimiento almacenado (con transacción):

Se ingresan como parámetros todos los datos necesarios para actualizar las tablas SecuenciaDeNucleotidos, Especie, Genero y Autor.


En realidad, nos interesa insertar en la tabla SecuenciaDeNucleotidos, pero si no existen la Especie, Genero y Autor ingresados, habrá que insertar previamente en esas tablas antes de poder insertar en SecuenciaDeNucleotidos. Para ello, existen IF anidados que se ejecutan según el caso: el primero se ejecuta en caso de que ya existan el autor y la especie ingresados, en cuyo caso no se inserta en esas tablas. La última cláusula del IF anidado se ejecuta en caso de que no existan ni la especie ni el autor, en cuyo caso deben ser insertados antes de insertar en la tabla SecuenciaDeNucleotidos. Para ello se emplea una transacción, que asegura de que se realicen las inserciones en las 4 tablas, y en caso de error en alguna de esas 4 inserciones, se revierte la transacción (rollback tran).

Observación 1: para simplificar el ejemplo, se asume que la Familia ingresada como parámetro y todos los taxones superiores siempre van existir en la base de datos, dado que los taxones superiores al nivel de familia no cambian tanto y no se describen nuevos de estos taxones frecuentemente.

Observación 2: en este caso se emplearon 4 temas técnicos para simplificar el proceso de inserción en las tablas: un procedimiento, 2 funciones, una vista y una transacción incluidos dentro del procedimiento. Si bien se simplifica la escritura a la hora de insertar, sería un trabajo posterior verificar cómo mejorar la eficiencia de ejecución.

Ahora, para insertar en 4 tablas, ingresamos los datos como parámetros del procedimiento, lo cual nos simplifica y abstrae de la complejidad de implementación.

```
EXEC SP_insertarSecuencia7 'ATATTTGT', '20220114', 'Maxam', null, 'Núcleo', 7, 'Lineal', 'Bufonidae', 'Ansonia', 'echinata', 'Esteban', 'Manolacc', 25389123
```

 Messages

(1 row affected)

Completion time: 2022-11-05T18:49:24.0910875-03:00

Ejecutamos la vista para verificar:

```
SELECT * FROM vw_vistaSecNucleotidos
```

Results		Messages							
	idSecNucleot	secuencia	fechaRegistro	metodo_secuenciacion	organela	conformacion	nroCrom...	Especie	anotaciones
12	13	TGTCCGCTTGGTATATTTTA...	2016-07-15	Método Enzimático de Sanger	Núcleo	Lineal	2	Rhinella marina	no
13	14	CGGGCCTTACATATGTG	2018-02-15	Método Enzimático de Sanger	Núcleo	Lineal	3	Rhinella abei	no
14	15	AGAGGTTAGATTGACCG	2020-03-01	Método Enzimático de Sanger	Núcleo	Lineal	2	Rhinella alata	no
15	16	AGAGGTTAGATTGACCG	2019-08-23	Método Enzimático de Sanger	Mitocondria	Circular	NULL	Rhinella alata	no
16	17	ATTGGTGTGGTAAACCG	2017-04-17	Método Enzimático de Sanger	Núcleo	Lineal	5	Rhinella icterica	no
17	21	ATATTTGT	2022-01-14	Método Químico de Maxam y Gilbert	Núcleo	Lineal	7	Ansonia echinata	NULL

Se insertó el registro correctamente.

ÍNDICES

Las tablas MetodoDeSecuenciacion, Organela y Conformacion presentan muy pocos registros (2 a 4), y además la situación de tales registros es prácticamente inamovible según el modelo de datos, por lo que no tendrán índices no agrupados.

Los siguientes índices no agrupados se agregarán a las demás tablas para mejorar la búsqueda por el campo indexado:

```
CREATE NONCLUSTERED INDEX IX_Gen_nombre
```

```
ON Gen(nombre);
```

```
CREATE NONCLUSTERED INDEX IX_Especie_epEsp
```

```
ON Especie(epitetoEspecifico);
```

```
CREATE NONCLUSTERED INDEX IX_Genero_nombre
```

```
ON Genero(nombre);
```

```
CREATE NONCLUSTERED INDEX IX_Familia_nombre
```

```
ON Familia(nombre);
```

```
CREATE NONCLUSTERED INDEX IX_Publicacion_tituloPublicacion
```

```
ON Publicacion(tituloPublicacion);
```

De esta manera se agiliza la búsqueda de secuencias de nucleótidos que pertenezcan a determinada Familia, o a determinado Género o Especie.

También podemos agregar índices a las vistas, por ejemplo en la vista vw_especies_taxon, agregaremos un índice no agrupado, pero para ello debemos modificar la definición de la vista con la cláusula **WITH SCHEMABINDING**, para que la definición de la tabla y de la vista queden ligados:

```
ALTER VIEW vw_especies_taxon
```

```
WITH SCHEMABINDING
```

```
AS
```

```
SELECT dom.nombre 'Dominio', r.nombre 'Reino', fil.nombre 'Filo', c.nombre  
'Clase',
```

```
o.nombre 'Orden', f.nombre 'Familia', g.nombre 'Género', sp.epitetoEspecifico  
'Epíteto específico',
```

```
sp.idEspecie, sp.idGenero, sp.idFamilia, sp.idOrden, sp.idClase, sp.idFilo,  
sp.idReino, sp.idDominio
```

```
FROM dbo.Especie sp INNER JOIN dbo.Genero g ON (sp.idGenero = g.idGenero)
```

```
INNER JOIN dbo.Familia f ON (g.idFamilia = f.idFamilia)
```

```
INNER JOIN dbo.Orden o ON (f.idOrden = o.idOrden)
```

```
INNER JOIN dbo.Clase c ON (c.idClase = o.idClase)
```

```
INNER JOIN dbo.Filo fil ON (c.idFilo = fil.idFilo)
```

```
INNER JOIN dbo.Reino r ON (fil.idReino = r.idReino)
```

```
INNER JOIN dbo.Dominio dom ON (r.idDominio = dom.idDominio)
```


Luego agregamos un índice agrupado y un índice no agrupado a la vista:

```
CREATE UNIQUE CLUSTERED INDEX PK_SP
ON vw_especies_taxon(idEspecie, idGenero, idFamilia, idOrden, idClase,
idFilo, idReino, idDominio);

CREATE NONCLUSTERED INDEX IX_VWespecies_especie
ON vw_especies_taxon([Género], [Epíteto específico]);
```

El último nos permite agilizar las consultas del siguiente tipo:

```
SELECT * FROM vw_especies_taxon
WHERE [Género] = 'Rhinella' AND [Epíteto específico] = 'marina'
```

PERMISOS

El esquema de permisos que plantearemos en nuestra base de datos consistirá en 3 roles definidos por el usuario y el rol predefinido db_owner. El rol db_owner será otorgado al DBA. A continuación, se detalla el código de los roles definidos por el usuario implementados, y luego realiza una explicación de los mismos:

```
USE db_genetica;
CREATE ROLE InvestigadorInterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.MetodoDeSecuenciacion to
InvestigadorInterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Organela to
InvestigadorInterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Conformacion to
InvestigadorInterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Titulo to InvestigadorInterno;
DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Pais to InvestigadorInterno;

GRANT SELECT on dbo.MetodoDeSecuenciacion to InvestigadorInterno;
GRANT SELECT on dbo.Conformacion to InvestigadorInterno;
GRANT SELECT on dbo.Organela to InvestigadorInterno;
GRANT SELECT on dbo.Titulo to InvestigadorInterno;
GRANT SELECT on dbo.Pais to InvestigadorInterno;

GRANT SELECT, INSERT on dbo.SecuenciaDeNucleotidos to InvestigadorInterno;
```

```

GRANT SELECT, INSERT on dbo.Autor to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Revista_Cientifica to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Publicacion to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Autor_Publicacion to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Secuencia_Autor to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Secuencia_Publicacion to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Gen to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Secuencia_Gen to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Secuencia_Gen_Potencial to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Especie to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Genero to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Familia to InvestigadorInterno;
GRANT SELECT, INSERT on dbo.Orden to InvestigadorInterno;
--Al investigador se le permite insertar hasta el nivel de Orden. Los niveles
superiores no suelen cambiar.
GRANT SELECT on dbo.vw_especies_taxon to InvestigadorInterno;
GRANT SELECT on dbo.vw_vistaSecNucleotidos to InvestigadorInterno;
GRANT EXEC on SP_insertarSecuencia to InvestigadorInterno;
-----

CREATE ROLE Curador;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.MetodoDeSecuenciacion to
Curador;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Organela to Curador;
DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Conformacion to Curador;

GRANT UPDATE, DELETE on dbo.SecuenciaDeNucleotidos to Curador;
GRANT UPDATE, DELETE on dbo.Gen to Curador;
GRANT DELETE on dbo.Secuencia_Gen to Curador;
GRANT DELETE on dbo.Secuencia_Gen_Potencial to Curador;
GRANT UPDATE, DELETE on dbo.Autor to Curador;
GRANT INSERT, SELECT, UPDATE, DELETE on dbo.Titulo to Curador;
GRANT UPDATE, DELETE on dbo.Revista_Cientifica to Curador;
GRANT UPDATE, DELETE on dbo.Publicacion to Curador;
GRANT UPDATE, DELETE on dbo.Especie to Curador;

```

```

GRANT UPDATE, DELETE on dbo.Genero to Curador;
GRANT UPDATE, DELETE on dbo.Familia to Curador;
GRANT UPDATE, DELETE on dbo.Orden to Curador;

-----

CREATE ROLE ConsultorExterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.MetodoDeSecuenciacion to
ConsultorExterno;

DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Organela to ConsultorExterno;
DENY UPDATE, INSERT, DELETE, CONTROL, ALTER on dbo.Conformacion to
ConsultorExterno;

GRANT SELECT on dbo.vw_autor to ConsultorExterno;

--Al consultor externo sólo se le permite ver la vista de Autor, la cual oculta
el campo DNI. No puede
--ver la tabla Autor

GRANT SELECT on dbo.MetodoDeSecuenciacion to ConsultorExterno;
GRANT SELECT on dbo.Conformacion to ConsultorExterno;
GRANT SELECT on dbo.Organela to ConsultorExterno;
GRANT SELECT on dbo.Titulo to ConsultorExterno;
GRANT SELECT on dbo.Pais to ConsultorExterno;
GRANT SELECT on dbo.SecuenciaDeNucleotidos to ConsultorExterno;
GRANT SELECT on dbo.Revista_Cientifica to ConsultorExterno;
GRANT SELECT on dbo.Publicacion to ConsultorExterno;
GRANT SELECT on dbo.Autor_Publicacion to ConsultorExterno;
GRANT SELECT on dbo.Secuencia_Autor to ConsultorExterno;
GRANT SELECT on dbo.Secuencia_Publicacion to ConsultorExterno;
GRANT SELECT on dbo.Gen to ConsultorExterno;
GRANT SELECT on dbo.Secuencia_Gen to ConsultorExterno;
GRANT SELECT on dbo.Secuencia_Gen_Potencial to ConsultorExterno;
GRANT SELECT on dbo.Especie to ConsultorExterno;
GRANT SELECT on dbo.Genero to ConsultorExterno;
GRANT SELECT on dbo.Familia to ConsultorExterno;
GRANT SELECT on dbo.Orden to ConsultorExterno;

```

A grandes rasgos, todos los roles, aparte del db_owner, tendrán denegadas las instrucciones del tipo DDL sobre las tablas y tampoco podrán actualizar, eliminar o insertar en las tablas Organela, Conformacion y MetodoDeSecuenciacion, dado que los registros de tales tablas no están previstos para ser modificados (excepto que se introduzca un nuevo método de secuenciación, por ejemplo, lo cual no ocurre frecuentemente, en cuyo caso podría ser realizado por el DBA).

El rol 'InvestigadorInterno' será el rol asignado a los investigadores que pertenecen a la organización, quienes podrán solamente insertar y consultar las tablas (no pueden modificar ni eliminar). Solo podrán insertar hasta el nivel taxonómico de Orden, dado que no son frecuentes los cambios en taxones superiores al nivel de Orden. También podrán acceder a procedimientos almacenados y vistas.

El rol 'Curador' es quien se encargará de mantener la integridad de los registros de la base de datos en caso de necesitarse modificar o eliminar registros en las principales tablas (excepto Organela, Conformacion, MétodoDeSecuenciacion). Es considerado un rol con mayor privilegio al cual pueden acudir los investigadores en caso de cometer un error en la inserción de datos.

El rol 'ConsultorExterno' será otorgado a usuarios que consulten la base de datos a través de internet, por ejemplo, a través de un sitio web perteneciente a la organización. Solo podrán acceder al **SELECT** de la gran mayoría de las tablas. No podrán acceder directamente a la tabla Autor, sino a través de la vista vw_autor, que oculta los campos dni, idMaxTitulo y descrEspecialidad, siendo dni un dato sensible que debe ocultarse al acceso externo. También podría crearse un procedimiento al cual accede este rol, que permita comparar una secuencia de nucleótidos ingresada con todos los datos de la base de datos, por ejemplo, el cual tendría una estructura muy similar a los procedimientos ya desarrollados.

Una vez definidos los roles, procederemos a crear logins y usuarios, y asignarles roles.

```
CREATE LOGIN investigador1 with password = '1234';
CREATE USER investigador1 for LOGIN investigador1;
CREATE LOGIN curador1 with password = '1234'
CREATE USER curador1 for LOGIN curador1;
CREATE LOGIN consultor1 with password = '1234'
CREATE USER consultor1 for LOGIN consultor1;

ALTER ROLE InvestigadorInterno ADD MEMBER investigador1;
ALTER ROLE Curador ADD MEMBER curador1;
ALTER ROLE ConsultorExterno ADD MEMBER consultor1;
```

De esta manera, hemos creado el usuario investigador1 y permitimos acceder al mismo mediante el login investigador1. Luego, le asignamos el rol 'InvestigadorInterno' al usuario investigador1. Y así sucesivamente pueden crearse nuevos inicios de

sesión asignados a un usuario que puede asignarse a uno o varios roles.

BACKUP

Un backup (copia de seguridad) es un duplicado de los datos que se hace para poder recuperarlos ante cualquier pérdida o incidente. Forman una parte muy importante de la seguridad en las bases de datos, ya que, sin ellas, se podrían perder infinidad de datos importantes.

SQL Server tiene 3 tipos de backups posibles:

- Backup Full: backup completo de la base de datos. Soportado en todos los modos de recovery de la base de datos.
 - Realizar un backup full todos los días no es recomendable, debido a que se ocupará mucho espacio en disco.
- Backup Diferencial: Es la diferencia entre el ultimo backup full y el momento en el que se desea hacer un nuevo backup. También está soportando en todos los esquemas de recovery en la base de datos. No es incremental.
 - El diferencial es más pequeño en tamaño en comparación del full. Si en lugar de utilizar el backup diferencial cuando se puede, utilizamos el backup full estaríamos desaprovechando espacio en disco y tiempo en backups. Utilizar el backup diferencial nos va a permitir ser mucho más óptimos.
- Backup TLOG (transaction log): Es un backup incremental. Es decir que para restaurar el TLOG 2 necesitamos el TLOG 1. Se recomienda utilizar esto en base de datos productivas. Solo está soportado en los modos de recovery full o bulk. El backup TLOG realiza una copia de seguridad de todas las transacciones que se hayan producido desde la última copia de seguridad o truncamiento del registro.
 - Los backups se pueden guardar afuera del SQL server, como en un recurso compartido, un blob storage de Azure, etc.
 - Una mala política de backup puede hacer perder mucho dinero a la organización y además poner en riesgo nuestro puesto de trabajo.

RECOVERY

Un recovery (recuperación) es el proceso por el cual se restauran los datos en una base de datos a partir de un backup.

Existen tres modelos de recuperación: simple, full (completa) y bulk (registros de operaciones masivas). Normalmente, en las bases de datos se usa el modelo de recuperación full o el modelo de recuperación simple.

Para definir el modelo de recuperación se debe ver el tipo de base de datos. No es lo mismo un servidor productivo que un servidor de testing.

Tipo de base de datos	Recovery aconsejable
Transaccional Producción	Full o Bulk
Transaccional desarrollo o testing	Simple
DW Producción	Simple
DW Desarrollo	Simple

Tabla 5. Recoveries recomendados por tipo de base de datos

Opciones para armar un plan de backup

- Planes de mantenimiento de Wizard de Microsoft (SSMS Wizard):
 - Es poco personalizable.
 - Simple de implementar.
 - Es recomendado para un DBA con poca experiencia.
- Herramientas de terceros:
 - Poco personalizable.
 - Depende de la herramienta para hacer los restore de una simple base de datos.
 - En algunos casos pueden generar problemas de performance en los MSSQL.
 - Puede haber perdidas de performance al realizar backups.
- Scripts vía Jobs:
 - Muy personalizable.
 - Se pueden aprovechar todas las opciones de backup y restore del motor.
 - Nativo de MSSQL sin la necesidad de usar terceras partes.
 - Es muy entendible para cualquier DBA.

El plan de backup que se utilizará para el proyecto son los Scripts vía Jobs: OLA Hallengren.

Scripts OLA Hallengren: Es la solución de mantenimiento de SQL Server más conocidos en la implementación de backup. Son muy completos y nativos en TSQL y además son gratuitos.

Para descargar los scripts OLA:

- Ir a la página oficial <https://ola.hallengren.com>
- Ir a la sección SQL Server Backup
- Descargar MaintenanceSolution.Sql

Para utilizar los scripts OLA:

- Activar SQL Server Agent en SSMS
- Ejecutar los scripts dentro de MaintenanceSolution.Sql

Una vez ejecutado los scripts, nos implementará los Jobs dentro del SQL Server Agent

Los Jobs instalados que más usaremos para los backups son:

SYSTEM_DATABASES – FULL: Backup full para las bases de datos del sistema

USER_DATABASES – DIFF: backup de tipo Diferencial

USER_DATABASES – FULL: backup de tipo Full

USER_DATABASES – LOG: backup de tipo TLOG

Para configurar los backups se deben editar los steps (pasos) y los schedules (horarios)

Para editar los steps de un Job se debe:

- ir a propiedades del Job
- Ir a steps
- Click en Edit

Algunas de las propiedades que se pueden editar son:

@Databases = 'SYSTEM_DATABASES', (En qué base de datos realizar el backup)

@Directory = NULL, (Dónde quiero guardar el backup)

@BackupType = 'FULL', (Tipo de backup)

@CleanupTime = NULL, (Cada cuánto tiempo en horas limpiar el backup anterior)

@Checksum = 'Y', (Verifica si la base de datos no está corrupta antes del backup)

Existen muchas más propiedades para modificar, las cuales se encuentran explicadas en la página de OLA.

Para programar el backup

Ir a propiedades

Ir a Schedules

Click en New

Ajustar los parámetros deseados

Nuestra estrategia de backup constará de:

- Un backup full una vez por semana, todos los domingos a las 00:00hs.
- Seis backups diferenciales por semana, de lunes a sábados a las 00:00hs de cada día.
- Backups TLOG cada 15 minutos de lunes a sábados. Lo que dará un total de 4 Backups tlog por hora, y 96 backups tlog por día. Al no haber tantas

transacciones como en una base de datos de un banco, mercados o similares, cada 15 minutos es un tiempo óptimo para evitar que el tamaño de cada tlog crezca y permitir la protección de los datos.



Figura 5. Esquema de backup propuesto para el proyecto

Demostración:

Colocamos el modo recovery de la base de datos en modo BULK, esto permite que todas las transacciones se almacenen en el LOG. De modo que la realización de un backup de LOG permitirá llevarse las transacciones antiguas a un fichero de Backup, y de este modo permitir reutilizar dicho espacio del TLOG (evitando que este tenga que seguir creciendo para poder almacenar las nuevas transacciones)


```
ALTER DATABASE [bd_genetica]
SET RECOVERY BULK_LOGGED WITH NO_WAIT
GO
```

- Eliminamos la base de datos para simular un error y proceder con las restauraciones

```
USE [master]
DROP DATABASE IF EXISTS [bd_genetica]
GO
```


- Para restaurar un TLOG primero hay que restaurar el backup full que lo contenga, y los anteriores backups TLOG en modo norecovery (esto es necesario ya que los backups TLOG son incrementales) y posteriormente el TLOG deseado en modo recovery.

```
/*Restauramos el backup FULL en modo norecovery*/
Restore database bd_genetica
from disk = 'C:\sql\backups\DESKTOP-
JSAR8LJ\bd_genetica\FULL\bd_genetica_FULL_20221114_200948.bak'
--norecovery para que la base de datos se mantenga en estado de restauración
with norecovery
```


 bd_genetica (Restoring...)


```
/*Luego restauramos el backup TLOG anterior al que queremos restaurar, también en norecovery*/
```

```
Restore log bd_genetica
from disk = 'C:\sql\backups\DESKTOP-
JSAR8LJ\bd_genetica\LOG\bd_genetica_LOG_20221114_200958.trn'
with norecovery
```

 bd_genetica (Restoring...)

```
/*y finalmente el backup tlog deseado, esta vez en modo recovery*/
```


```
Restore log bd_genetica
from disk = 'C:\sql\backups\DESKTOP-
JSAR8LJ\bd_genetica\LOG\bd_genetica_LOG_20221114_202001.trn'
with recovery
```

 bd_genetica

- Para restaurar un Backup Diferencial, similar al TLOG. Primero hay que restaurar el backup full que lo contenga en modo norecovery, y posteriormente el Backup Diferencial correspondiente


```
/*Restauramos el backup FULL en modo norecovery*/
```

```
Restore database bd_genetica
from disk = 'C:\sql\backups\DESKTOP-
JSAR8LJ\bd_genetica\FULL\bd_genetica_FULL_20221114_200948.bak'
with norecovery
```

 bd_genetica (Restoring...)

```
/*Y luego restauramos el backup diferencial correspondiente al backup full anterior */
```

```
Restore log bd_genetica
from disk = 'C:\sql\backups\DESKTOP-
JSAR8LJ\bd_genetica\DIFF\bd_genetica_DIFF_20221114_201004.bak'
with recovery
```

 bd_genetica

- En ambos casos, de no haber ningún error, la base de datos ya estaría funcional para su uso

Manejo de archivos multimedia en Bases de datos

Muchas veces necesitamos guardar archivos multimedia (imágenes, audios, videos, archivos gif, pdf, words, etc.) en nuestra base de datos. Para ello hay diversas opciones:

- Utilizando Varbinary(max): Es una forma de almacenar archivos dentro de la base de datos. Utilizando este tipo de dato (Varbinary(max)) podemos lograr almacenar un archivo con una previa conversión a ese tipo de dato. Esta es una de las formas más clásicas y viejas que tiene SQL server de almacenar multimedia dentro de la base de datos.
 - Varbinary se recomienda para los que quieran guardar únicamente las imágenes en SQL

Demostración:

Creación de una tabla que contiene un campo "imagenPortada" con un tipo de dato varbinary

```
CREATE TABLE Publicacion(  
    idPublicacion int identity(1,1) not null,  
    tituloPublicacion varchar(255) not null,  
    imagenPortada varbinary(MAX),  
    extension char(4)  
);
```

GO

Insertión de un registro

```
INSERT INTO [dbo].[ Publicacion] ([tituloPublicacion], [imagenPortada],  
[extension])  
SELECT 'Aves', BULKCOLUMN, 'JPG'  
FROM OPENROWSET(BULK N'C:\Users\juanx\Desktop\Aves\Pajaro.jpg', SINGLE_BLOB) AS  
DATA
```

GO

OPENROWSET permite leer datos de muchas fuentes. BULK tiene una gran capacidad para leer datos de forma individual, cuando se especifica un objeto BLOB, este puede leer el archivo de formas distintas. En este caso se utilizó SINGLE_BLOB para que lea el archivo como Varbinary(max)

- Guardando el nombre del archivo: Otra forma de guardar archivos en la base de datos es simplemente a través de un campo varchar(), donde se guarde el nombre del archivo junto con su extensión como una cadena común y corriente. Luego cuando se desee utilizar el archivo en alguna aplicación, se deberá concatenar la ruta donde se encuentre el archivo con su respectivo nombre.
 - Es una buena manera de tratar con archivos, debido a que la base de datos solo se encarga de guardar el nombre o el directorio/nombre del archivo, ahorrando mucho espacio en la misma.

Demostración:

Creación de una tabla que contiene un campo “NombreImagenPortada” con un tipo de dato varchar()

```
CREATE TABLE Publicacion(  
    idPublicacion int identity(1,1) not null,  
    tituloPublicacion varchar(255) not null,  
    NombreImagenPortada varchar(100)  
);  
  
GO
```

Insertión de un registro

```
INSERT INTO [dbo].[ Publicacion] ([tituloPublicacion], [NombreImagen])  
VALUES ('Familia de Abejas Colletidae' , 'Colletidae.jpg')  
  
GO
```

- FileStream: Es una funcionalidad que nos permite utilizar el tipo de dato Varbinary almacenado fuera del SQL Server, pero estando totalmente integrado al mismo. Por ejemplo, si realizamos un backup de la base de datos, se estarán incluyendo todos los archivos dentro del FileStream.
 - Es una muy buena forma de trabajar con grandes volúmenes de archivos, teniendo al propio archivo como un campo en una tabla, pero fuera de la base de datos.

Antes de utilizar FileStream se debe activarlo a nivel motor, para ello se deben seguir los siguientes pasos:

- Ir a administrador de configuración de SQL server
- Desplegar SQL server services
- Click derecho en SQL server
- Ir a Propiedades
- Ir a Pestaña FILESTREAM y Activar “Enable FileStream to TransactSQL e I/O”
- Colocar el nombre del recurso compartido
- Ejecutar el siguiente script en SSMS:

```
EXEC sp_configure filestream_access_level 2 RECONFIGURE
```

- Por último, reiniciar el servicio de SQL server

Demostración:

Creación de una tabla para el manejo de archivos

```
CREATE TABLE Archivo_fs (  
    idArchivo int identity(1,1),  
    id2 UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE DEFAULT NEWID(),
```

```

        titulo varchar(50) not null,
        contenido VARBINARY(MAX) FILESTREAM,
        extension CHAR(4),

        CONSTRAINT UK_id2_Archivo UNIQUE(id2),
        CONSTRAINT PK_Archivo_fs PRIMARY KEY(idArchivo)
    );

```

La función NEWID() inicializa el valor del tipo de dato UNIQUEIDENTIFIER

Inserción de un archivo a la tabla

```

INSERT INTO [dbo].[Archivo_fs] ([titulo], [contenido], [extension])
SELECT 'DER', BULKCOLUMN, 'PNG'
FROM OPENROWSET(BULK N'C:\Users\juanx\Desktop\Archivos\DER.PNG', SINGLE_BLOB) AS
DATA

```

Al tratarse de un tipo de dato varbinary(max), la inserción de nuevos registros es similar al primer caso.

- FileTable: Es una tabla de usuario especializada con un esquema predefinido que almacena los datos FileStream, así como información de jerarquía de directorios y archivos e información de atributos de archivos. Al igual que en FileStream, los datos se almacenan fuera del SQL Server, pero están totalmente integrado al mismo. Soporta la creación de subcarpetas y nombres de archivos personalizados.

Se pueden usar estos métodos para cargar archivos en una FileTable

- Arrastrar y colocar los archivos desde las carpetas de origen hasta la nueva carpeta de FileTable del explorador de Windows
- Usar opciones de la línea de comandos como MOVE, COPY, XCOPY o ROBOCOPY, desde el CMD o en un archivo o script por lotes
- Escribir una aplicación personalizada para mover o copiar los archivos en c# o VB .NET

Por dentro utiliza una tecnología similar al FileStream, por lo que se requiere de la misma configuración hecha anteriormente del FileStream más lo siguiente:

- Establecer el nombre del directorio
- Colocar la propiedad non_transacted_acces en full

Esto se puede realizar con el siguiente script:

```

USE [master]
GO
ALTER DATABASE [bd_genetica]
SET FILESTREAM( NON_TRANSACTED_ACCESS = FULL,
DIRECTORY_NAME = N'BDGenetica' )
WITH rollback immediate
GO

```

Rollback immediate desconecta a los usuarios de la base de datos instantáneamente

- Verificamos los valores de las propiedades non_transacted_access, non_transacted_access_desc y si existe el directorio en la base de datos seleccionada con:

```
SELECT DB_NAME (database_id) as 'DB Name', directory_name, non_transacted_access,
non_transacted_access_desc
FROM sys.database_filestream_options
GO
```

- Creamos la tabla FileTable

```
USE [bd_genetica]
```

```
drop table if exists PublicacionArchivos_FT
```

```
CREATE TABLE PublicacionArchivos_FT AS FileTable
WITH
(
    FileTable_Directory = 'PublicacionArchivos_FT',
    FileTable_Collate_Filename = database_default,
    FILETABLE_STREAMID_UNIQUE_CONSTRAINT_NAME=UQ_steam_id
);
GO
```

FileTable_Directory: Especifica el directorio raíz para los archivos relacionados al FileTable creado

FileTable_Collate_Filename: Especifica el nombre de la intercalación que se aplicará a la columna Name

database_default: La columna hereda la intercalación de la base de datos actual

(Intercalación: proporcionan propiedades de distinción entre mayúsculas y minúsculas, acentos y reglas de ordenación para los datos)

Formas de utilizar FileTable:

- Ver todos los atributos del directorio

```
select * from [dbo].[PublicacionArchivos_FT]
GO
```

- Actualizar atributos de cada archivo mediante T-SQL

```
update [dbo].[PublicacionArchivos_FT]
set is_readonly = 0
where name = 'DER.png'
GO
```

- Eliminar archivos mediante T-SQL

```
delete
from [dbo].[PublicacionArchivos_FT]
where name = 'DER2.png'
GO
```

- Ver ubicación de los archivos en el directorio utilizando el tipo de dato HierarchyID

```
SELECT [path_locator] as FileLocation, [parent_path_locator] as ParentFolder
FROM [dbo].[PublicacionArchivos_FT]
GO
```

- Ver ubicación de los archivos en el directorio utilizando el tipo de dato HierarchyID convertido a string (resulta más legible para el DBA)

```
SELECT [path_locator].ToString() as FileLocation,
[parent_path_locator].ToString() as ParentFolder
FROM [dbo].[PublicacionArchivos_FT]
GO
```

- Creación de subdirectorios mediante T-SQL

```
INSERT INTO PublicacionArchivos_FT(Name, is_directory) values('Imagenes', 1)
INSERT INTO PublicacionArchivos_FT(Name, is_directory) values('Videos', 1)
INSERT INTO PublicacionArchivos_FT(Name, is_directory) values('Audios', 1)
INSERT INTO PublicacionArchivos_FT(Name, is_directory) values('Archivos', 1)
```

- Creación de una vista convirtiendo los tipos de datos HierarchyID en varchar() (Necesario por si se desea trabajar con Entity Framework en una aplicación, debido a que EF no reconoce el tipo de dato HierarchyID)

```
CREATE VIEW [dbo].[VistaPublicacionArchivos_FT]
AS
SELECT [stream_id]
      ,[file_stream]
      ,[name]
      --dependiendo del directorio y subdirectorios a usar, el tamaño del
      varchar puede variar, ser más pequeño o más grande acorde a las necesidades
      ,[CONVERT(VARCHAR(4000),[path_locator]) AS [path_locator]
      ,[CONVERT(VARCHAR(4000),[parent_path_locator]) AS [parent_path_locator]
      ,[file_type]
      ,[cached_file_size]
      ,[creation_time]
      ,[last_write_time]
      ,[last_access_time]
      ,[is_directory]
      ,[is_offline]
      ,[is_hidden]
      ,[is_readonly]
      ,[is_archive]
      ,[is_system]
      ,[is_temporary]
FROM [dbo].[PublicacionArchivos_FT]
;
```

- Deshabilitar FileTable

```
ALTER TABLE [dbo].[PublicacionArchivos_FT]
DISABLE FILETABLE_NAMESPACE
```

GO

- Eliminar FileTable

```
DELETE FROM [dbo].[PublicacionArchivos_FT]
GO
```

- Habilitar FileTable

```
ALTER TABLE [dbo].[PublicacionArchivos_FT]
ENABLE FILETABLE_NAMESPACE
GO
```

- Ejemplo de inserción de archivos mediante XCOPY Y ROBOCOPY desde la terminal de Windows (CMD)

XCOPY (ruta del archivo o directorio a copiar) (ruta de destino):

```
xcopy C:\Users\juanx\Desktop\Lithobates_palustris_(rana).jpeg \\Desktop-
jsar8lj\mssqlserver\bd_genetica_fs\PublicacionArchivos_FT
```

ROBOCOPY (ruta del directorio de archivos) (ruta de destino):

```
robocopy C:\Users\juanx\Desktop\archivosbd\imagenes \\Desktop-
jsar8lj\mssqlserver\bd_genetica_fs\PublicacionArchivos_FT /copy:d
```

CAPITULO V: CONCLUSIONES

A partir de la investigación realizada y el desarrollo del trabajo, se intentó modelar e implementar los requerimientos provenientes de una disciplina científica (genética), conectándolos con los temas técnicos de las bases de datos informáticas para su implementación. Pese a las complejidades de conectar estas dos disciplinas, creemos que se lograron los objetivos planteados, al menos desde una óptica introductoria a ambas disciplinas, lo cual abre un panorama interesante para idear otro tipo de implementaciones más complejas y eficientes, o de crear un modelado mucho más enriquecido de conceptos genéticos que, por brevedad y también por falta de conocimientos mucho más profundos de la genética como disciplina, fueron omitidos.

CAPTITULO VI: BIBLIOGRAFÍA

- **Beaulieu, A.** (2009). *Learning SQL: master SQL fundamentals*. O'Reilly Media, Inc..
- **Elmasri, R., Navathe, S. B., Castillo, V. C., Pérez, G. Z., & Espiga, B. G.** (2007). *Fundamentos de sistemas de bases de datos* (No. QA76. 9D3 E553 2007.). Pearson educación.
- **Posada, D.** (Ed.). (2009). *Bioinformatics for DNA sequence analysis*. New York: Humana Press.