Del Esquema Conceptual al Mundo Real, una Base de Datos Genómica

Tesis de Máster

Luis Eduardo Eraso Schattka

Máster en Ingeniería de Software, Métodos Formales y Sistemas de Información

Del Esquema Conceptual al Mundo Real, una Base de Datos Genómica

Tesina presentada por:
Luis Eduardo Eraso Schattka

Supervisores:
Laura Mota Herranz
Oscar Pastor López

Valencia, España. Febrero 2011

Agradecimientos

En primer lugar agradezco a Dios, por permitir llegar a donde estoy, por brindarme la oportunidad de realizar este máster y finalizar este trabajo.

Agradezco a mis padres, Eduardo y Patricia, por ser mis compañeros incondicionales y mis mejores amigos, por su constante apoyo, sus inagotables consejos y su inmenso amor y apoyo que me han brindado durante toda mi vida y en especial, durante la recta final de este trabajo.

Agradezco a toda mi familia, que desde la distancia sé que me envían sus mejores deseos.

A mis amigos, Samuel, Alex y Elkin, por estar siempre cuando más los necesitaba.

Agradezco a Oscar Pastor López, por brindarme su conocimiento y experiencia, además por permitir realizar esta tesina con él y el grupo de investigación Genoma.

Agradezco muy especialmente a Laura Mota Herranz, por darme la chispa y la fuerza en el momento que más lo necesitaba, por sus valiosas aportaciones y el tiempo dedicado, pues sin ella no hubiese sido posible la consecución de este trabajo.

Índice general

Índice de figuras

Capítulo 1

Introducción

Existen muchas técnicas para el diseño y desarrollo de sistemas en la actualidad, sin embargo, el uso de técnicas de modelado conceptual esta siendo una práctica cada vez más común entre la ingeniera de software, ya que con esta técnica se proporciona una precisa descripción del dominio del problema, de tal forma que la obtención de modelos a diferentes niveles de abstracción determine el producto software final.

Así pues el Modelado Conceptual consiste en entender y dominar dominio del problema y la conceptualización del conocimiento que se tiene sobre él, a un nivel abstracto antes de implementar una solución, permite que los ingenieros de software y los clientes de los sistemas trabajen al mismo nivel y que, además, exista un entendimiento sobre lo que se tiene que hacer y el producto que se obtendrá.

El diseño de modelos conceptuales y llevarlos a aplicaciones reales es abordado a través de la ingeniera dirigida por Modelos MDE (Model-Driven Engineering). Estas técnicas de modelado conceptual han sido aplicadas en los últimos años a muchos dominios de sistemas de información como sistemas de negocios, sistemas médicos, sistemas financieros, etc., sin embargo, se desea llevar estas técnicas a niveles más extremos, a dominios más difíciles y desafiantes como lo es la interpretación del Genoma Humano, donde

la ausencia del uso de las técnicas del modelado conceptual es notable, y la falta de conocimiento que se tiene sobre el dominio es poca.

La Bioinformática es una disciplina científica emergente que utiliza la tecnología de la información para organizar, analizar y distribuir información biológica con la finalidad de responder preguntas complejas en biología. Además es un área de investigación multidisciplinaria, pues abarca dos ciencias: Biología y Computación. Sin embargo, la evolución del campo de la Bioinformática parece estar más orientada al diseño de gran de algoritmos de búsqueda o de alineaciones de cadenas de ADN más eficientes, en lugar de aplicar las buenas prácticas que la Ingeniera de software ofrece para el desarrollo de sus sistemas.

Los principales objetivos de aplicar el Modelado Conceptual a la interpretación del Genoma Humano son:

1. Modelar correctamente el conocimiento que se tiene sobre este dominio y
2. Construir un sistema de información basado en dicho modelo conceptual.

El trabajo que se presenta en esta tesina se va a centra más en el segundo objetivo pues, tanto para la Genómica como para la Bioinformática es importante desarrollar sistemas de software de calidad que contribuyan a la interpretación e investigación en este dominio.

El trabajo se ha desarrollado dentro del Centro de Investigación ProS (Métodos de Producción del Software) en concreto en la línea de investigación de Modelado Conceptual del Genoma y, junto con otras tesinas desarrolladas en el centro [5, 14, 15, 16], tiene como objetivo diseñar una plataforma de base de datos que incluya toda la información relevante que se pueda encontrar en las distintas bases de datos existentes así como el desarrollo de aplicaciones que permita su explotación.

Estos objetivos generales, se concretan en los siguientes subobjetivos:

1. Diseño de un Esquema Conceptual del Genoma Humano (ECGH) que represente todo el dominio del conocimiento[ref].

2. Diseño de la base de datos relacional asociada mediante un proceso de transformación y refinamiento.

3. Análisis de las fuentes de datos existentes y más utilizadas por los biólogos.

4. Diseño de un prototipo de carga inicial que permita poblar la base de datos definida.

5. Implementación y puesta en marcha del prototipo.

Los subobjetivos cubiertos por la tesina que se presenta son el segundo (capítulo 4), el tercero (capítulo 5) y el cuarto (capítulo 6). En el capítulo 2 se presenta el ECGH, en el capítulo 3 se hace una revisión al estado del arte de los trabajos relacionados y que han servido de punto de partida al proyecto realizado. La memoria concluye con las conclusiones en el capítulo 7.

Capítulo 2

Sistemas de Información y Informática: Un Modelo Conceptual para el Genoma Humano

Resumen: En este capítulo se presenta una introducción al Modelado Conceptual, cuáles son sus objetivos y ventajas de uso, de igual forma se hace una introducción al dominio de la Bioinformática y se explica el porqué y qué ventajas conlleva aplicar el modelado conceptual a un campo tan complejo como la Bioinformática.

El Modelado Conceptual, es la actividad que tiene como objetivo obtener y definir conocimiento sobre un sistema. En este contexto, debe entenderse por sistema un conjunto de elementos adecuadamente relacionados entre sí y con su entorno. Los sistemas pueden ser naturales, filosóficos, económicos, de información, etc.

Modelar conceptualmente un sistema implica la especificación de un dominio, es decir, la identificación de las propiedades relevantes del sistema, en

un instante determinado, así como su comportamiento. Estas propiedades relevantes son una abstracción de la realidad según el punto de vista del observador.

Cada sistema tiene una serie de objetivos, que deben ser tomados en cuenta cuando se realiza la descripción de sus características relevantes. Los sistemas de información (SI) son sistemas que contribuyen a que otros sistemas (más amplios) cumplan sus objetivos.

Para entender la constitución y el funcionamiento del genoma humano y poder modelarlo conceptualmente, es fundamental tener un panorama general de lo que se considera Biología Molecular.

La Biología Molecular es el estudio de la vida a un nivel molecular. Esta área está relacionada con otros campos de la Biología y la Química, particularmente la Genética y la Bioquímica. La Biología Molecular concierne principalmente al entendimiento de las interacciones de los diferentes sistemas de la célula, lo que incluye muchísimas relaciones, entre ellas las del ADN con el ARN, la síntesis de proteínas, el metabolismo, y el cómo todas esas interacciones son reguladas para conseguir un afinado funcionamiento de la célula. Al estudiar el comportamiento biológico de las moléculas que componen las células vivas, la Biología Molecular roza otras ciencias que abordan temas similares. Para el caso concreto de la elaboración de este trabajo de investigación, se habla de la Genética que se interesa por la estructura y funcionamiento de los genes y por la regulación de la síntesis intracelular de enzimas y de otras proteínas.

Los dominios a los que se han aplicado las técnicas de modelado conceptual son muchos, y probablemente el ámbito de aplicación más conocido y trabajado sea el relacionado con los Sistemas Organizacionales. Cuando se exploran nuevos dominios de aplicación, como la Bioinformática, aparecen nuevos desafíos que dependen de la complejidad del dominio. El dominio del Genoma Humano es uno de estos dominios, en él llama la atención el hecho de que no se hayan aprovechado las ventajas que ofrece el modelado conceptual, para conseguir su correcta y completa especificación.

Los conceptos biológicos en este dominio son descritos por medio de un esquema conceptual que permita un mejor entendimiento del genoma humano: las relaciones estructurales y funcionales de los genes con el proceso de traducción del ADN y los procesos de transcripción implicados en la síntesis de proteínas. Tradicionalmente, los desarrollos en el campo de la Bioinformática han estado orientados a la resolución de problemas algorítmicos y computacionales, subestimando la importancia que tiene para el área la existencia de sistemas de información genómicos que sean fiables y que estén preparados para asumir los continuos cambios a los que está sometido este campo de investigación.

El uso de las técnicas de modelado conceptual en el desarrollo de software moderno permite tener una descripción más exacta del problema del dominio, además la aplicación de dichas técnicas antes de desarrollar implementaciones garantiza que el software desarrollado cumplan los requisitos de calidad deseados. Sin embargo, estos principios que han guiado el diseño e implementación del desarrollo de software no se tienen muy en cuenta en dominios como la Bioinformática. Por tal motivo, en este trabajo se explica cómo se ve el dominio del Genoma Humano desde un punto de vista de Sistema de Información y cómo la interpretación del Genoma Humano puede ser afrontada desde una perspectiva de modelado conceptual con la intención de describir y entender más a fondo y con más claridad este dominio tan complejo, para así, una vez se tenga el esquema conceptual estable, proceder a la creación de Sistemas de Información Biológicos estables y fiables teniendo en cuenta los principios de desarrollo de software por modelos. Teniendo en cuenta las ventajas del modelado conceptual, se define un esquema conceptual donde se representan los conceptos básicos del genoma humano utilizados por los biólogos cuando abordan procesos relacionados con los análisis genéticos. En este esquema la descripción realizada tiene la intención de identificar los conceptos relevantes que están involucrados en la estructura y funcionamiento del organismo humano, desde el ADN hasta la producción de proteínas que mantienen la estructura y actividad celular en el organismo humano.

Este Esquema Conceptual del Genoma Humano (ECGH), el cual ha sido

elaborado por el grupo Genoma del Centro de Investigación en Métodos de Producción Software (ProS) del Universidad Politécnica de Valencia [13], corresponde con un estado intermedio y estable del conocimiento actual sobre el dominio. El ECGH se describe usando el estándar UML concretamente se han utilizado los diagramas de clase como lenguaje de modelado. El Esquema Conceptual se muestra dividido en sus tres vistas (vista gen-Mutation, vista Transcription y vista Genome) para su mejor visualización (Ver Fig. 2.1, Fig. 2.2 y Fig. 2.3).

La forma especial de identificar clases de conceptos en este dominio, supone considerar que todos los conceptos evolucionan continuamente. Esta evolución de conceptos se debe a los avances que van sucediendo en la investigación en Biología Molecular. Por lo tanto, al especificar conceptualmente el genoma humano, ha sido muy importante tener en cuenta que el esquema experimenta una constante evolución. Dicha evolución está relacionada con la información que se adquiere a partir de un conocimiento más profundo del genoma humano, y también está determinada por la naturaleza evolutiva de los elementos del dominio. Una completa descripción del esquema conceptual, su evolución, la incorporación de nuevos conceptos y restricciones de integridad están descritas en [5].

El ECGH y la creación de la base de datos, puede considerarse una herramienta eficiente para el diseño e implementación de un Sistema de Información que contribuya de manera eficiente a la resolución de los problemas de integración de datos para la búsqueda y recuperación de información valiosa de los estudios realizados sobre la secuenciación de genomas de los seres humanos.

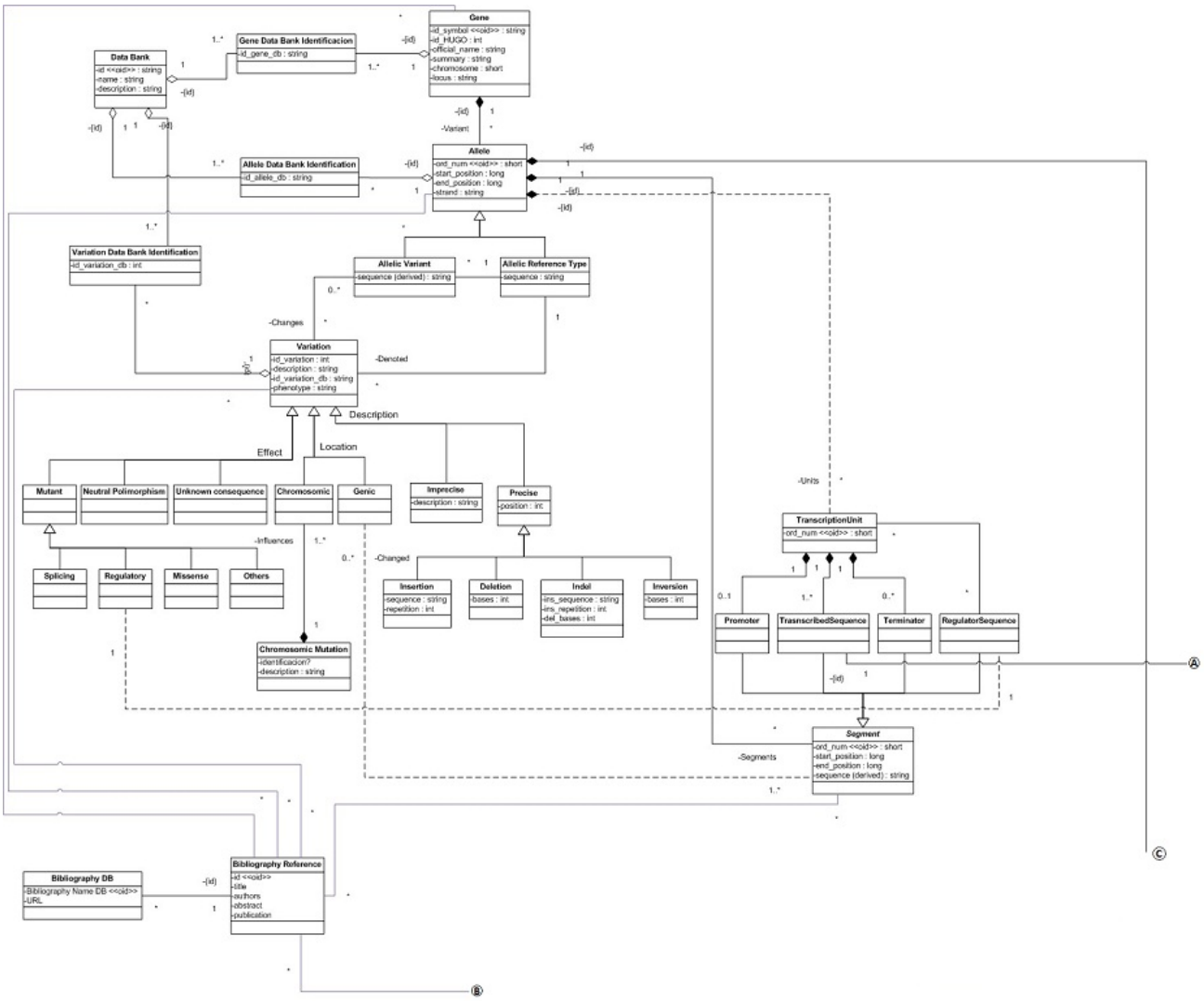


Figura 2.1: Vista Gene-Mutation del ECGH

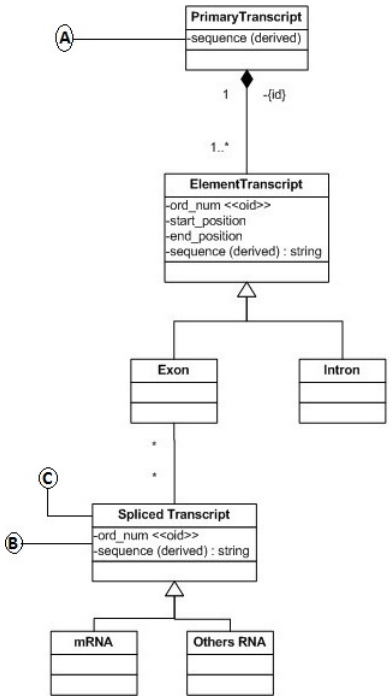


Figura 2.2: Vista Transcription del ECGH

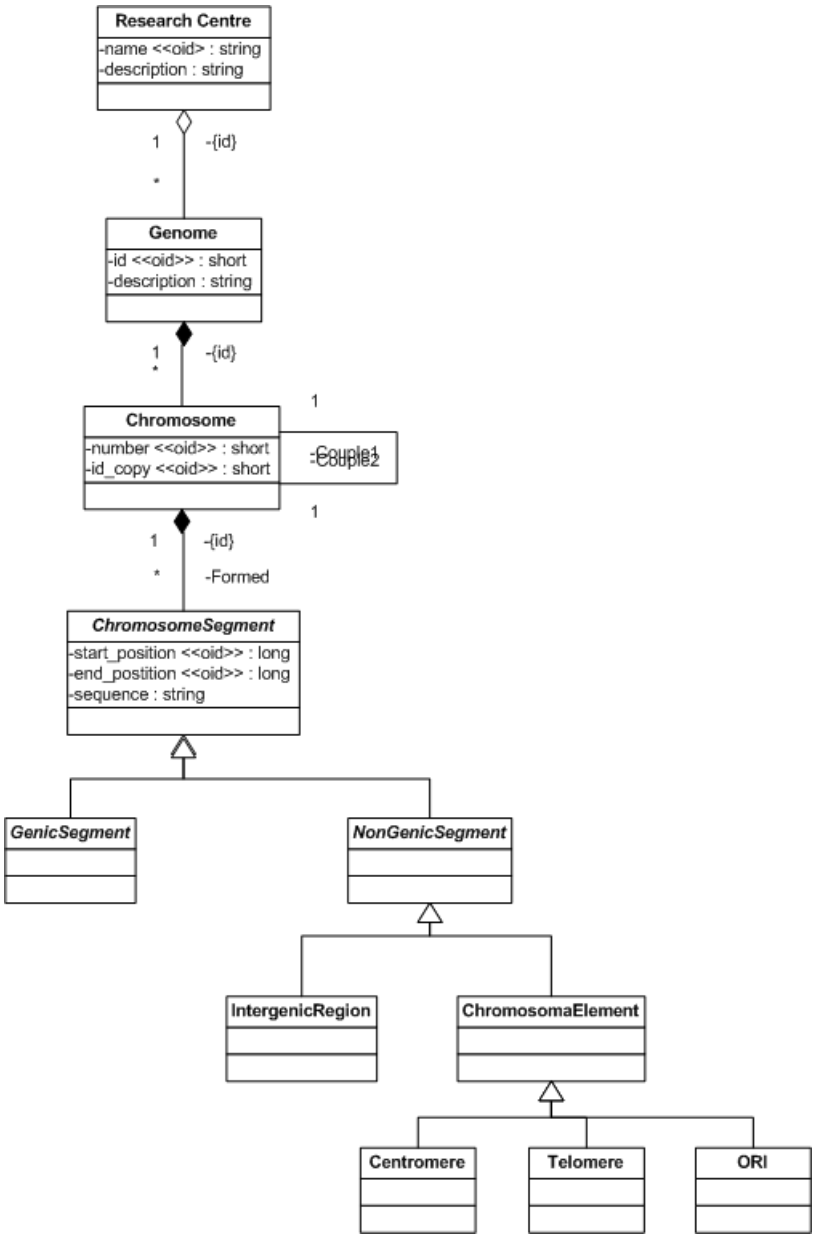


Figura 2.3: Vista Genome del ECGH

Este esquema debe cumplir las siguientes restricciones de integridad:

- Una variante de alelo se relaciona con un alelo tipo de referencia de su mismo gen.

- Los segmentos se relacionan con unidad de transcripción de su mismo alelo.
- Los tránscritos se relacionan con exones de su mismo alelo.

Capítulo 3

Trabajos Relacionados

Resumen: En este capítulo se presentan los trabajos que más se relacionan y que han servido de punto de partida para la realización de este proyecto.

En [1] los autores proponen una aproximación sistemática para el modelado, la captura y diseminación de los datos experimentales proteómicos, ya que la generación y el análisis de estos datos así como las técnicas y la tecnología están en una constante evolución. Los autores presentan una aproximación de un esquema conceptual en UML del Repositorio de Datos Proteómicos Experimentales (PEDRO). A partir de dicho esquema, describen implementaciones desarrolladas en XML y SQL, y discuten estrategias de captura, almacenamiento y diseminación de los datos proteómicos. Este trabajo facilita el desarrollo de herramientas de búsqueda más efectivas previniendo la información ambigua que existe en las diferentes bases de datos existentes. Su principal objetivo es capturar toda la información relevante de los experimentos proteómicos: los detalles de los experimentos, la muestra origen, los métodos y equipamientos utilizados, los resultados y los análisis. Para cumplir con sus objetivos se ha realizado un esquema, según los autores, con un alto grado de flexibilidad ya que la tecnología en este campo está en una evolución continua y el correspondiente repositorio debe de anticipar y eventualmente acomodar los datos generados por un nuevo tipo de experimento.

El esquema de PEDRO representa un subconjunto de toda la información sobre los experimentos proteómicos, sin embargo, los autores creen que es suficiente para forjar una base para la cual se desarrollan los correspondientes repositorios y herramientas para almacenar, mantener y consultar dicha información. Se plantean también diferentes ventajas a partir de la adopción del modelo. Para el investigador, todos los conjuntos de datos tendrán la información suficiente para establecer la procedencia y relevancia del conjunto de datos, además de permitir búsquedas no estándares. Se podrán desarrollar herramientas que permitan el acceso a un número grande de conjunto de datos además de facilitar el intercambio de información entre investigadores. Este trabajo se presenta con el objetivo de ver cómo, a partir del Modelado Conceptual se conceptualiza toda la información que aborda un dominio. Luego una vez esté establecido dicho esquema, se procede a transformarlo a un esquema relacional (creación de los repositorios) y enseguida al desarrollo de herramientas de mantenimiento y explotación de la información capturada. Este trabajo tiene gran relación con el trabajo de esta tesis pues ambos empiezan modelando conceptualmente un dominio específico. Sin embargo, una diferencia muy clara que existe es que en dicho trabajo se ha realizado un modelado muy genérico del dominio sin tener en cuenta si la información que se necesita realmente es posible encontrarla, es decir, no existe un nivel preciso de detalle de los datos, dejando así, este esquema más orientado a un dominio ideal que a un dominio real. Mientras que el trabajo desarrollado en esta tesis, sí que tiene en cuenta la realidad del dominio, realizando un análisis del estado actual de la información, dando como resultado un de Modelo Real al igual que su instancia (Esquema Relacional y Bases de Datos).

Otro trabajo que se relaciona con esta tesis es [2] . Los autores plantean un almacén de datos llamado Atlas, donde se almacena y se integra localmente secuencias biológicas, interacciones moleculares, anotaciones funcionales de genes y ontologías biológicas. Su objetivo es brindar una infraestructura de datos para desarrollos e investigaciones en el campo de la bioinformática. Los autores definen que uno de los objetivos primordiales en el campo de la informática es el de integrar los datos de las diferentes bases de datos fuente

heterogéneas que se utilizan en este dominio, ya que la mayoría de los repositorios biológicos se enfocan y proveen un particular tipo de dato. Esto puede permitir a los investigadores descubrir nuevas asociaciones entre los datos y validar las hipótesis existentes. Además, para los bioinformáticos el trabajar con la información pública existente es una tarea difícil que requiere un gran esfuerzo, ya que dicha información está en constante crecimiento tanto cuantitativamente como en complejidad. Los autores han dado una solución a esta problemática, creando un almacén de datos con herramientas de explotación y mantenimiento de información. Primero, los autores han establecido aproximaciones usando el Modelo Relacional de Datos. Las bases de datos creadas a partir de dichos modelos serán pobladas por los datos fuente, y luego podrán ser consultadas y explotadas usando herramientas desarrolladas para tal fin. La arquitectura que desarrollaron para este sistema consta de tres principales capas: la capa de Datos Fuente, la capa de Bases de Datos, y la capa de Recuperación. Los datos integrados en el sistema son descargados primero como archivos de datos de las bases de datos fuente. Estos archivos de datos son mapeados y cargados en las bases de datos del sistema a través de cargadores construidos para tal fin. La capa de Base de Datos del sistema está dividida en cuatro grupos, según tema biológico. Estos grupos son: Secuencias, Interacciones Moleculares, Genes y Catalogación Funcional y Ontologías. En la capa de Recuperación, para cada base de datos del sistema existe un método de recuperación de datos construidos en SQL, C++, JAVA y Perl. La capa de Recuperación es flexible y da una interfaz accesible para las bases de datos. Los datos pueden ser usados usando un cliente MySQL, a través de las API's, o las herramientas *end – user* construidas. De este trabajo cabe resaltar que se ha construido, a partir del Modelo Relacional, un sistema para proporcionar acceso de alto rendimiento y de forma flexible a la información biológica de diferentes bases de datos fuente, permitiendo a los biólogos y científicos de la computación llevar a cabo consultas para sus investigaciones. Además, se establece una arquitectura por capas, dando flexibilidad e independencia al sistema, tanto a nivel de recuperación como a nivel de explotación. Dicha arquitectura es un modelo a tener en cuenta para la definición de la arquitectura del sistema que se plantea como trabajos

futuros en el capítulo 6 de esta tesis. Sin embargo, se ve claramente que lo que se hace en este proyecto es definir un Esquema Relacional instanciado de las diferentes bases de datos según temas biológicos, simplemente con el objetivo de replicar la información localmente de las diferentes bases de datos que existen en el dominio y definir mappings entre éstas. Además, no existe una conceptualización del dominio reflejado en un modelo, a diferencia del trabajo realizado en esta tesis. La solución que se plantea en el trabajo [2] abarca de forma muy general y no es una solución absoluta a los problemas que los bioinformáticos experimentan día a día, por el hecho de brindar toda la información existente, mas no se hace un verdadero análisis de qué información sí es necesaria y cuál no, como sí se hace en esta tesis al realizar un modelado conceptual y verificar que dicha conceptualización se ajuste al estado actual del dominio.

Otro trabajo interesante, y de gran relación al trabajo realizado en esta tesis es [3]. La hipótesis inicial de este trabajo consistía en que para un total entendimiento de la función de un gen era necesaria la integración de diferentes conjuntos de datos. A partir de esta idea, los autores han desarrollado un trabajo que consta en dos partes, una base de datos que contiene diferentes datos genómicos, como secuencias de genomas, datos transcriptómicos, interacciones proteína – proteína, ontología de gen y caminos metabólicos, y por otro lado, un ambiente de análisis que soporta consultas complejas de selección y ejecución sobre los datos. Los autores para la realización de este trabajo se han basado en el esquema conceptual del genoma humano descrito por Paton et al. [4]. Luego se ha instanciado este esquema conceptual en un esquema de la base de datos, el cual está divido en cinco partes fundamentales que representan la secuencia del genoma, las interacciones proteína–proteína, los datos transcriptómicos, los caminos metabólicos y las ontologías del gen. El esquema conceptual en el que se han basado los autores para la realización de este trabajo, ha sido el esquema conceptual de partida para la realización del trabajo [5] y la continuación de esta tesis, sin embargo en [5], se realizan cambios claves y muy significativos al esquema propuesto por Paton, para extender las ideas iniciales, dar una mejor conceptualización y entendi-

miento del dominio con el objetivo principal proponer un completo esquema conceptual del genoma humano.

Capítulo 4

Diseño e Implementación de la Base de Datos Genómica

Resumen: En este capítulo se describe cómo se diseña y se implanta la base de datos Genómica a partir del Esquema Conceptual del Genoma Humano (ECGH). El capítulo se ha organizado como sigue: primero se realiza una introducción a los fundamentos teóricos del diseño de una base datos relacional desde un esquema conceptual en UML, para ello se introducirá qué es el Diagrama de Clases, sus elementos y funcionalidades, de igual forma se introducirá el Modelo Relacional de Datos. Una vez vistos estos conceptos, es necesario establecer reglas de transformación del Esquema Conceptual al Esquema Relacional, y, una vez aplicadas dichas reglas, realizar el correspondiente proceso de normalización y de optimización al Esquema Relacional resultante. Una vez se tiene el esquema Relacional optimizado, se procede a la implementación de la base de datos describiendo el proceso de diseño físico de la bases de datos en un servidor Oracle 10 g.

4.1. Fundamentos Teóricos: Diagrama de Clases y Modelo Relacional de Datos

A continuación se hace una fundamentación teorica del Diagrama de Clases para poder enteder su funcionalidad, denotación y respresentanción:

4.1.1. Diagrama de Clases

La parte de UML que está relacionada con la representación de los datos se denomina Diagrama de Clases. Por lo tanto es la que se va a emplear para representar el Esquema Conceptual del Genoma Humano, y es el lenguaje de modelado que se ha transformado en estándar en los últimos años para generar los modelos de datos de las aplicaciones. El UML está basado en el paradigma orientado a objetos y posee una serie de modelos que permiten plasmar con facilidad los requerimientos de una aplicación, su diseño e implementación.

El diagrama de clases es un tipo de diagrama que describe la estructura de un sistema mostrando los objetos de información que tiene, las propiedades de estos objetos y las conexiones existentes entre ellos. Un diagrama de clase, tiene dos elementos básicos, la **clase** y la **relación**. A continuación se presentan estos elementos con más detalle.

Clase: Son los elementos que permiten representar toda la información de un objeto del dominio. A través de ella podemos modelar el dominio en estudio. En UML, una clase se representa por un rectángulo que posee tres divisiones:

- **Parte Superior:** contiene el nombre de la Clase.
- **Parte Intermedia:** contiene los atributos que caracterizan a la Clase. Los atributos pueden representarse sólo mostrando su nombre, mostrando su nombre y su tipo, e incluso su valor por defecto. Tambien

se puede representar: los identificadores de la clase con la etiqueta <<oid>>, si el atributo puede tomar el valor no nulo, o si es un atributo con restriccion de unicidad.

- **Parte Inferior:** contiene los métodos u operaciones que definen cómo interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

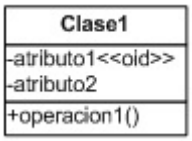


Figura 4.1: Ejemplo de una Clase

Relación: Son los elementos que permiten representar las conexiones entre clases. Pueden ser de tres tipos diferentes: Relación de Asociación, Relación de Dependencia y Relación de Generalización/Especialización. Las relaciones además tienen una propiedad importante denominada multiplicidad o cardinalidad que indica el nivel de dependencia entre las clases al especificar cuántas instancias de una clase se pueden relacionar con una sola instancia de otra clase. La cardinalidad se anota en cada extremo de la relación y los valores más frecuentes de cardinalidad son:

- 1: Un elemento relacionado.
- 0..1: Uno o ningún elemento relacionado.
- 0..*: Varios elementos relacionados o ninguno.
- 1..*: Varios elementos relacionados pero al menos uno.
- M..N Entre M y N elementos relacionados.

A continuación se presentan con más detalle los tres tipos de relaciones antes comentados:

- **Relación de Asociación:** este tipo de relación permite asociar objetos que colaboran entre sí. Una asociación describe la relación entre clases de objetos y describe posibles uniones, donde una unión es una instancia de una asociación, al igual que un objeto es una instancia de una clase. Una Relación de Asociación se representa con una línea sólida que une dos clases (Ver Fig. 4.2). El grado de una asociación se determina por el número de clases conectadas por la misma asociación así, pueden ser binarias, ternarias o de mayor grado. A su vez las relaciones de asociación pueden ser asociaciones reflexivas, es decir que relaciona distintos objetos de una misma clase (Ver Fig. 4.3).

Figura 4.2: Ejemplo de Relación de Asociación entre dos clases

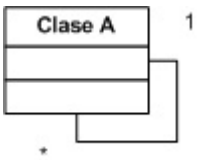


Figura 4.3: Ejemplo de Asociación Reflexiva

Para describir el papel que cada clase tiene en la asociación se puede hacer el uso de los roles. Un rol es una etiqueta que aparece en los extremos de la línea que denota la relación (Ver Fig. 4.4). Del ejemplo se puede deducir que muchas Personas *trabajan* en muchas Compañías.

Existen tres formas especiales de las asociaciones, la Asociación de Composición, la Asociación de Agregación y la Asociación por identificación.

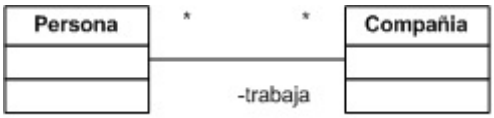


Figura 4.4: Ejemplo de una Asociación con roles

- **Asociación de Composición:** también conocidas como Asociaciones Fuertes. Esta asociación establece una relación entre la clase Padre y la clase Hijo, donde la clase hijo depende de las clases padre para su existencia. La dependencia es tan fuerte que implica que al eliminar la instancia padre, cualquier instancia hija será eliminada también. Las asociaciones de composición requiere que la clase padre tenga cardinalidad Uno (1). La Asociación de Composición se representa mediante una línea con un rombo sólido (Ver Fig. 4.5).

Figura 4.5: Ejemplo de Asociación de Composición

Del ejemplo de la figura 4.5, se deduce que un libro se compone de varios capítulos y un capítulo aparece en exactamente un libro. Si se elimina una instancia de la clase Libro, todas las instancias de la clase Capítulo relacionadas con ella también dejarán de existir debido a la asociación de composición entre las clases.

- **Asociación de Agregación:** también conocidas como Asociaciones Débiles. La Agregación implica que los miembros de la agrupación son independientes de la agregación misma. La notación usada para este tipo de asociación es una línea con un rombo vacío. El rombo vacío identifica la clase padre. Se usan las mismas cardinalidades que se usan en las asociaciones normales. Una clase agregada puede estar incluida en varias agregaciones simul-

táneamente, esto se debe a la debilidad de la relación, ya que esto permite que la clase hija sobreviva a la eliminación de sus clases padres.

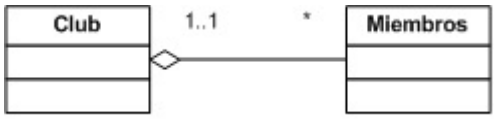


Figura 4.6: Ejemplo de Asociación de Agregación

En el figura 4.6, se ve el ejemplo de una asociación de agregación, donde se deduce que a un club pueden pertenecer varios miembros, sin embargo, si el Club se disuelve o deja de existir, los miembros siguen existiendo.

- **Asociación por Identificación**: Esta asociación implica que la identificación de una clase se consigue gracias a su asociación con otra clase. Se representa con la etiqueta {id} en el arco de la asociación.

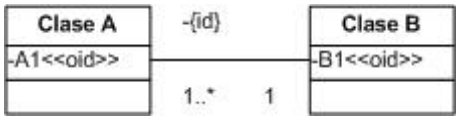


Figura 4.7: Ejemplo de Relación por Identificación

La clase B debe tener cardinalidad 1 en la asociación, mientras que la cardinalidad de A podrá ser * ó 1,en el primer caso B deberá tener al menos un atributo etiquetado con <<oid>>.

- **Relación de Dependencia:** Es una relación de uso, es decir una clase usa a otra, que la necesita para su cometido. Se representa con una flecha discontinua va desde la clase utilizadora a la clase utilizada. Con la dependencia mostramos que un cambio en la clase utilizada puede afectar al funcionamiento de la clase utiliza-

dora, pero no al contrario.

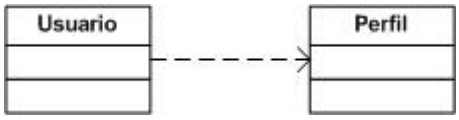


Figura 4.8: Ejemplo de Relación de Dependencia

- **Relación de Generalización/Especialización:** Las clases con atributos y operaciones comunes se pueden organizar de forma jerárquica, mediante la generalización, también conocida como herencia. La herencia es una abstracción importante para compartir similitudes entre clases, donde todos los atributos y operaciones comunes a varias clases se pueden compartir por medio de la superclase, una clase más general. Las clases más refinadas se conocen como subclases. La herencia es útil tanto para el modelado conceptual, al proporcionar una buena estructuración de los objetos, como para la proceso de implementación al evitar replicar innecesariamente código. La superclase generaliza a sus subclases, y las subclases especializan a la superclase. El proceso de especialización es el inverso de generalización. Una instancia de una subclase, o sea un objeto, es también una instancia de su superclase. La herencia indica que una subclase hereda los métodos y atributos especificados por una superclase, por ende la subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la superclase. Este tipo de relación se representa mediante una flecha que apunta a la clase que más abarca o a la clase más alta, es decir a la superclase (Ver Fig. 4.9)

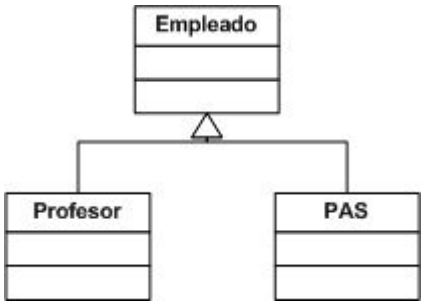


Figura 4.9: Ejemplo de Relación de Generalización

El diagrama de clases de UML incluyes algunos objetos mas pero dado que no se han utilizando en el modelado conceptal, no se explicarán en este trabajo.

4.1.2. Modelo Relacional de Datos

El Modelo Relacional es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos en el que se han basado algunos de los sistemas de gestión de bases de datos más relevantes hoy en día siendo por ello el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos. Fue propuesto por Edgar Frank Codd en 1970 [6], de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos, imponiéndose sobre los modelos anteriores (jerárquicos y red) dado a la sencillez, ya que el usuario percibe la base de datos como un conjunto de datos organizados en filas y en columnas. Además, la información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y potencia para administrar la información.

El Modelo Relacional proporciona dos estructuras de datos: la *Tupla* y la *Relación*. La estructura *Tupla* permite representar objetos del mundo real a través de sus propiedades (atributos). Un tipo tupla se define como un conjunto de pares de la forma {(A1, D1),(A2, D2), . . ., (An, Dn)} denominado esquema de la tupla, donde {A1, A2,. . ., An} es el conjunto de nombres

de los atributos y D1, D2,. . . , D3 son los dominios asociados de dichos atributos.

La estructura de datos *Relación*, permite representar el conjunto de ocurrencias de un mismo tipo de objeto. En un tipo relación se define un conjunto de pares de la forma: {(A1, D1), (A2, D2),. . . , (An, Dn)}, denominado esquema de la relación. Una relación de esquema {(A1, D1), (A2, D2),. . . , (An, Dn)} es un conjunto de tuplas de dicho esquema. Es importante darse cuenta que el esquema de una relación concuerda con el esquema de sus tuplas.

Al definir una relación como un conjunto de tuplas de un mismo esquema, se puede dar que las relaciones no representen estados válidos de los objetos que se está representando. Para evitar este tipo de problemas y para darle una mayor expresividad al Modelo Relacional, se agrega el concepto de *Restricción de Integridad* al modelo. Una *Restricción de Integridad* representa una propiedad que deben cumplir los datos para que sean considerados una instancia válida de la Base de Datos. Existen cuatro tipos de restricciones:

1. **Restricción de Valor No Nulo:** esta propiedad expresa que no debe haber en una relación una tupla que tenga el valor nulo en el atributo sobre el que se define la propiedad.

2. **Restricción de Unicidad:** esta propiedad expresa que no debe existir en una Relación dos tuplas que tengan el mismo valor en todos los atributos del conjunto de atributos sobre el que se define la propiedad.

3. **Restricción de Clave Primaria:** para facilitar la identificación y la manipulación de las relaciones en la base de datos, se introduce el concepto de Clave Primaria. Una Clave Primaria es un conjunto de atributos de su esquema, el cual es elegido como identificador de sus tuplas. La Clave Primaria siempre debe tener un valor para cada tupla, es decir, no podrá tener un valor nulo, además, el valor de la clave primaria deberá ser un valor único para cada tupla.

4. **Restricción de integridad Referencial:** para expresar asociaciones entre los objetos representados por las relaciones del esquema se introduce el concepto de Clave Ajena. La forma de expresar estas asociaciones consiste en incluir en el esquema de una relación R, el identificador de otra relación S, a este conjunto de atributos se les conoce como clave ajena de la relación R que hace referencia a la relación S.
   Además de conocer los atributos que constituyen la clave ajena y la relación referida por ella, se debe especificar las *directrices de borrado* y *actualización* asociadas a esa clave ajena, es decir, el comportamiento del sistema frente a actualizaciones de la base de datos que violen esa integridad referencial.
   Las directrices pueden ser de borrado o de modificación:

   a) Directrices de borrado. Definen el comportamiento del sistema ante el borrado de una tupla en una relación a la que se hace referencia desde otra relación con una clave ajena si este borrado supone la violación de la integridad referencial y puede ser:

      1) Borrado restrictivo: el sistema rechazará el borrado. (Directriz por defecto).
      2) Borrado a nulos: el sistema sustituirá en la clave ajena el valor desaparecido por el valor no nulo.
      3) Borrado en cascada: el sistema borrará en la relación que contiene la clave ajena todas las tuplas que apuntaran a la tupla borrada.

   b) Directrices de modificación. Definen el comportamiento del sistema ante la modificación del valor en la clave principal en una tupla de una relación a la que se hace referencia desde otra relación con una clave ajena si esta modificación supone la violación de la integridad referencial y puede ser:

      1) Modificación restrictiva: el sistema rechazará la modificación. (Directriz por defecto).

2) Modificación a nulos: el sistema sustituirá en la clave ajena el valor desaparecido por el valor no nulo.
3) Modificación en cascada: el sistema modificará en la relación que contiene la clave ajena todas las tuplas que apuntaran a la tupla modificada.

Teniendo en cuenta las anteriores restricciones de integridad, el concepto de tuplas y de relaciones, la notación a utilizar para representar el Esquema Relacional resultante de la transformación del Esquema Conceptual, proceso explicado en el punto 4.2 de este capítulo, será el siguiente:

$$
\begin{gathered}
R : (A_1 : D_1, A_2 : D_2, \ldots, A_r : D_r) \\
CP : \{A_1, \ldots, A_r\} \\
CAj : \{A_0, \ldots, A_p\} \rightarrow S \\
f\,(A_0) = B_j \\
\ldots \\
f\,(A_p) = B_n \\
\textit{Directriz de Borrado} \\
\textit{Directriz de Actualización} \\
S : (B_1 : E_1, B_2 : E_2, \ldots, B_t : E_t) \\
CP : \{B_1, \ldots, B_r\} \\
UNI : \{B_q, \ldots, B_r\} \\
VNN : \{B_s, \ldots, B_t\}
\end{gathered}
$$

Donde, $R : (A_1 : D_1, A_2 : D_2, \ldots, A_r : D_r)$ es la relación con los atributos y los correspondientes dominios de los atributos, $CP : \{A_1, \ldots, A_r\}$será la clave primaria de la relación, $CAj : \{A_0, \ldots, A_p\} \rightarrow S$ será una clave ajena que asocia una relación con otra, $UNI : \{B_q, \ldots, B_r\}$ representa la restricción de unicidad, y por último, $VNN : \{B_s, \ldots, B_t\}$ especifica los valores no nulos de la relación.

4.2. Transformación del Esquema Conceptual al Esquema Relacional de Datos

Una vez se ha definido el Esquema Conceptual del Genoma Humano (ECGH), se procede a realizar su correspondiente transformación al modelo de datos elegido, el cual en este caso es el Modelo Relacional. El resultado de esta transformación será el Esquema Relacional del Genoma Humano (MRGH) que se utiliza para crear la Base de Datos Genómica.

Estas transformaciones entre los esquemas son necesarias, pues así se podrá entender el porqué y el cómo han evolucionado los modelos a través de tiempo, debido a nuevas necesidades a nuevos conocimientos del dominio. Además, gracias a estas transformaciones, cada cambio realizado en el esquema conceptual, se verá reflejado en el modelo relacional y por lo tanto en la base de datos genómica asegurando de esa forma una completa correspondencia y actualización entre los esquemas.

En ocasiones el modelo relacional es menos expresivo que el modelo conceptual utilizado siendo necesario añadir algunas restricciones en lenguaje natural para suplir la falta de expresividad. Estas restricciones se añadirán al esquema relacional con una etiqueta numerada RIn (Restricción de Integridad).

4.2.1. Reglas de transformación del Esquema Conceptual al Esquema Relacional

La transformación del Esquema Conceptual al Esquema Relacional, no es una tarea fácil, sin embargo teniendo en cuenta las reglas de transformación del Modelo Entidad – Relación al Modelo Relacional descrito por [12] se pueden adaptar perfectamente a la transformación deseada de estos esquemas. Para esto se definen las siguientes reglas:

- **Transformar las Clases a Relaciones:** es decir, cada clase del modelo será traducida a una relación. A su vez, cada atributo de la clase será un atributo de la relación, Además cada relación tendrá como clave primaria el identificador de la clase, también es importante hacer un análisis de los dominios de los atributos y cómo serán transformados a los tipos de datos específicos del SGBD que se escoja. Las propiedades de Valor No Nulo y unicidad se trasladarán directamente de la clase al esquema relacional.

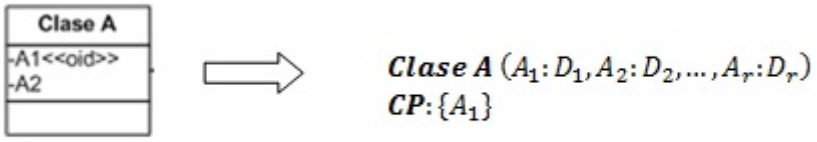


Figura 4.10: Ejemplo de transformación de una Clase a una Relación

- **Transformar las Relaciones entre las Clases**, se debe de analizar el tipo de relación de la clase, su cardinalidad y si existen o no restricciones de identificación, así, teniendo en cuenta dichas características se procede a transformar dicha relaciones de clases.

    1. **Relaciones de Asociación:** Para las relaciones de Asociación los posibles valores de cardinalidad son: Una a Una (1..1 : 1..1), Cero o Uno a Cero o Uno (0..1 : 0..1), Una a Muchas (1..1: N..N), Cero o Uno a Muchas (0..1 : N..N) ,muchas a muchas (N..N: N..N). Para cada valor de cardinalidad se define una regla de transformación:

        a) **Relación de asociación con cardinalidad Una a Una (1..1: 1..1):** Cada clase se transforma en una relación con clave principal el identificador de la clase correspondiente y alguna de las dos relaciones tendrá como clave ajena el identificador de la otra relación con la cual está relacionada, siendo

esta clave ajena un valor No Nulo y Único.

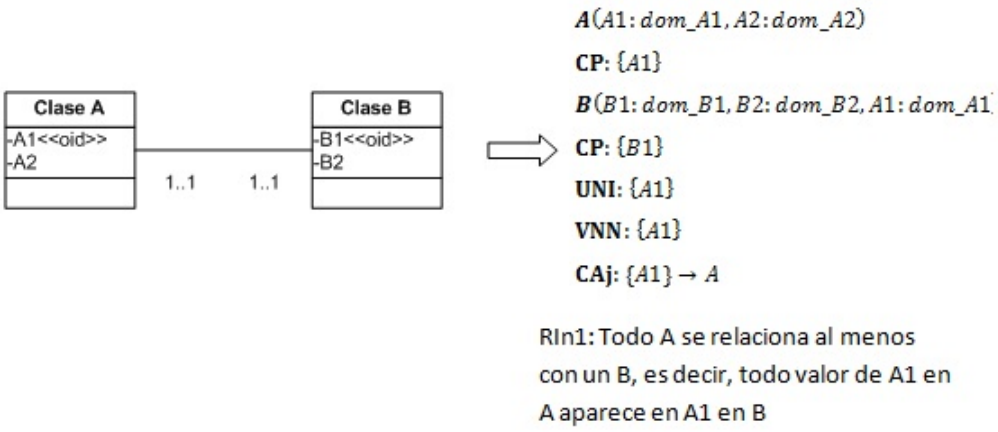


Figura 4.11: Ejemplo de transformación de una Relación de Asociación con cardinalidad Una a Una

b) **Relación de asociación con cardinalidad Cero o Uno a Cero o Uno (0..1 : 0..1):** Cada clase se transforma en una relación con clave principal el identificador de la clase correspondiente y alguna de las dos relaciones tendrá como clave ajena el identificador de la otra relación con la cual está relacionada, siendo esta clave ajena un valor Único.

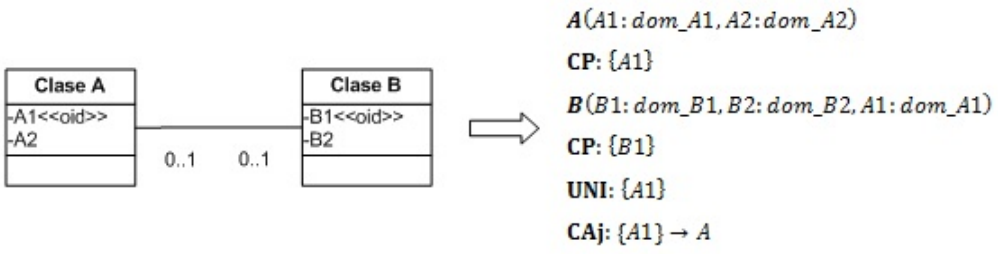


Figura 4.12: Ejemplo de transformación de una Relación de Asociación con cardinalidad Cero o Uno a Cero o Uno

c) **Relación de asociación con cardinalidad Una a Muchas (1..1 : N..N):** Cada clase se transforma en una relación con clave primaria el identificador de la clase correspondiente y la clave de la clase que participa con cardinalidad máxima Uno pasa como clave ajena de la otra relación. Teniendo esta clave ajena un valor No Nulo.

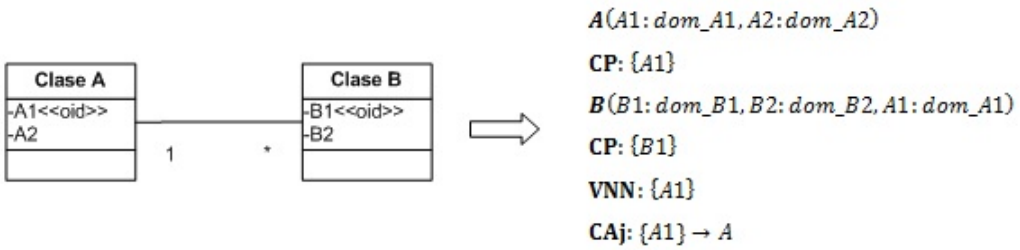


Figura 4.13: Ejemplo de transformación de una Relación de Asociación con cardinalidad Una a Muchas

d) **Relación de asociación con cardinalidad Cero o Uno a Muchas (0..1 : N..N):** Cada clase se transforma en una relación con clave primaria el identificador de la clase correspondiente y la clave de la clase que participa con cardinalidad máxima Uno pasa como clave ajena de la otra relación con la cual está relacionada.

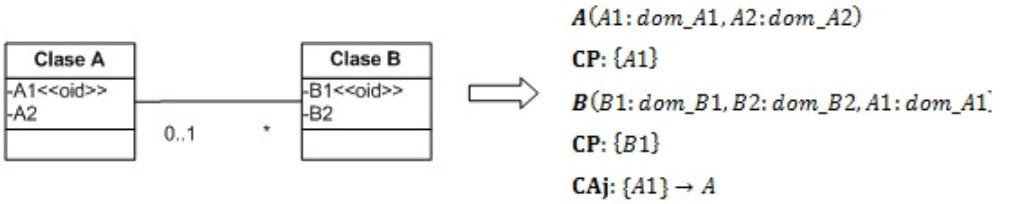


Figura 4.14: Ejemplo de transformación de una Relación de Asociación con cardinalidad Cero o Uno a Muchas

e) **Relación de asociación con cardinalidad Muchas a Muchas (N..N : M..M):** Cada clase se transforma en una relación con clave primaria el identificador de la clase correspondiente, además se construye una nueva relación correspondiente a la asociación, cuya clave primaria estará formada por la unión de los identificadores de las clases que participan en la asociación.

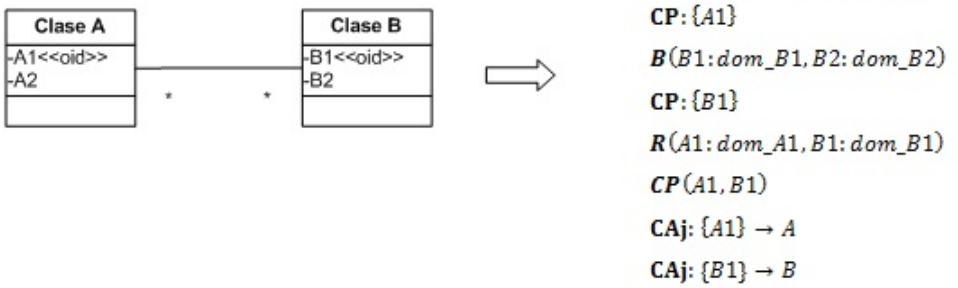


Figura 4.15: Ejemplo de transformación de una Relación de Asociación con cardinalidad Muchas a Muchas

f) **Asociación de composición:** se deben transformar de las misma forma que una asociación de cardinalidad de Uno a Muchos, esto se debe a la condición de que la clase padre tenga cardinalidad Uno (1). En este caso la directriz de restauración de la integridad referencial de la clave ajena será de "Borrado en Cascada" para asegurar que cuando desaparece la clase padre el sistema también borrará las clases hijas.

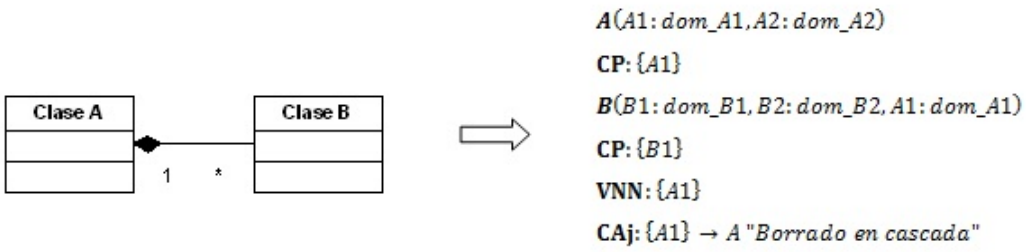


Figura 4.16: Ejemplo de transformación de una Asociación de composición

g) **Asociación de Agregación:** en las asociaciones de agregación los miembros de la agregación son independientes de la propia agrupación, por esta razón este tipo de asociación se debe de tratar de la misma forma como una asociación.

h) **Asociación por Identificación**: La transformación de una clase que tiene una o varias asociaciones por identificación es análoga a la de una clase normal con la diferencia de que es necesario incorporar, como atributos, las claves primarias de las tablas de las otras clases participantes en esas asociaciones. La clave principal de la clase será la unión de las claves principales incluidas junto con los atributos propios etiquetados como $<<oid>>$ (si es que existen).

Para ilustrar esta transformación se incluyen varios ejemplos:

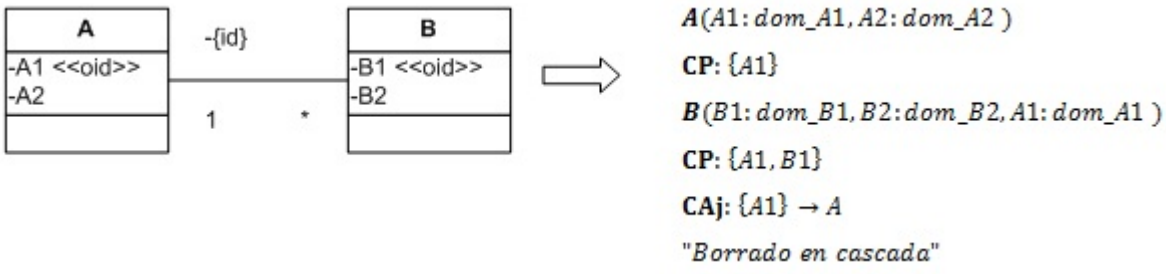


Figura 4.17: Ejemplo de Transformación de una Asociación por identificación

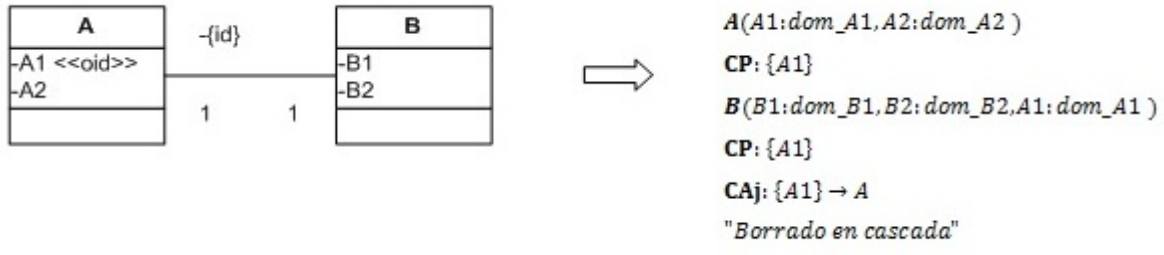


Figura 4.18: Ejemplo de Transformación de una Asociación por identificación

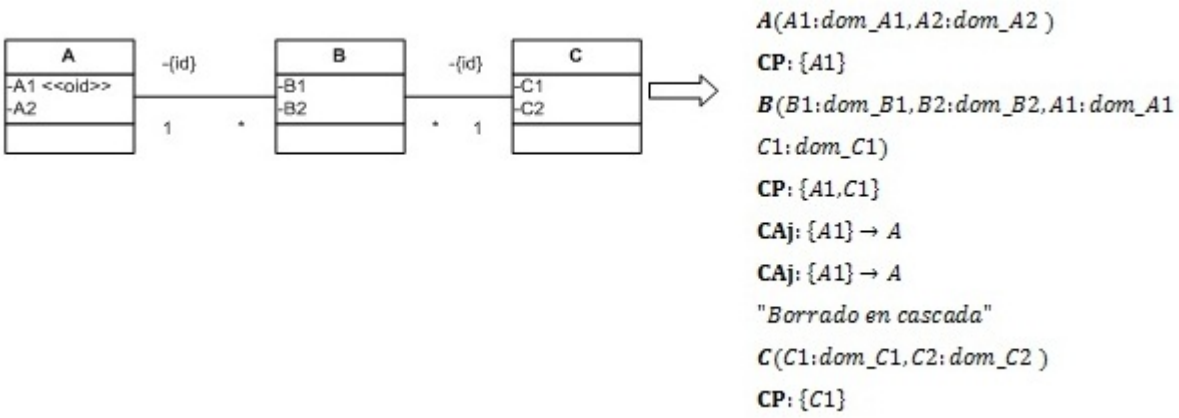


Figura 4.19: Ejemplo de Transformación de una Asociación por identificación

2. **Relación de Generalización/Especialización:** La transformación consiste en definir una tabla para la superclase como una clase normal y una tabla para cada subclase que incluye los atributos propios y también la clave primaria de la tabla de la superclase como clave ajena. En la tabla de la subclase, la clave ajena incluida es también la clave primaria.

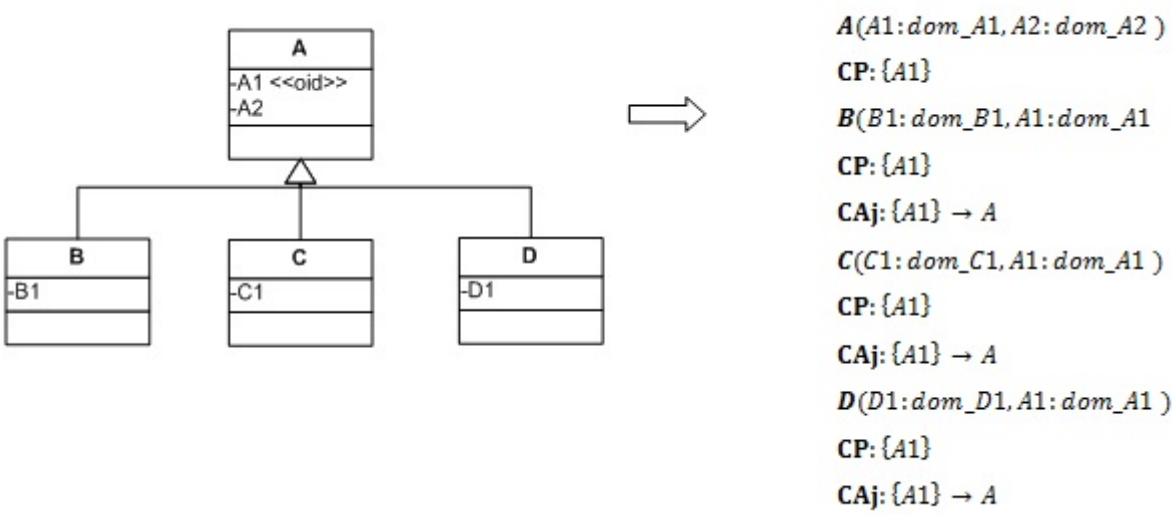


Figura 4.20: Ejemplo de Transformación Relación de Generalización/Especialización

A continuacion se procede a explicar la transformación del ECGH aplicando las reglas anteriormente descritas.

4.3. Transformación del Esquema Conceptual al Esquema Relacional del Genoma Humano

Una vez se han definido la reglas de transformación se procede a aplicarlas sobre el Esquema Conceptual del Genoma Humano, dicho esquema está dividió en tres vistas: la vista Gene-Mutation (Ver Fig. 4.21), la vista Transcription y la vista Genome, de las cuales, las vistas Gene-Mutation y la vista Transcription serán las que se traducirán al Modelo Relacional del Genoma Humano.

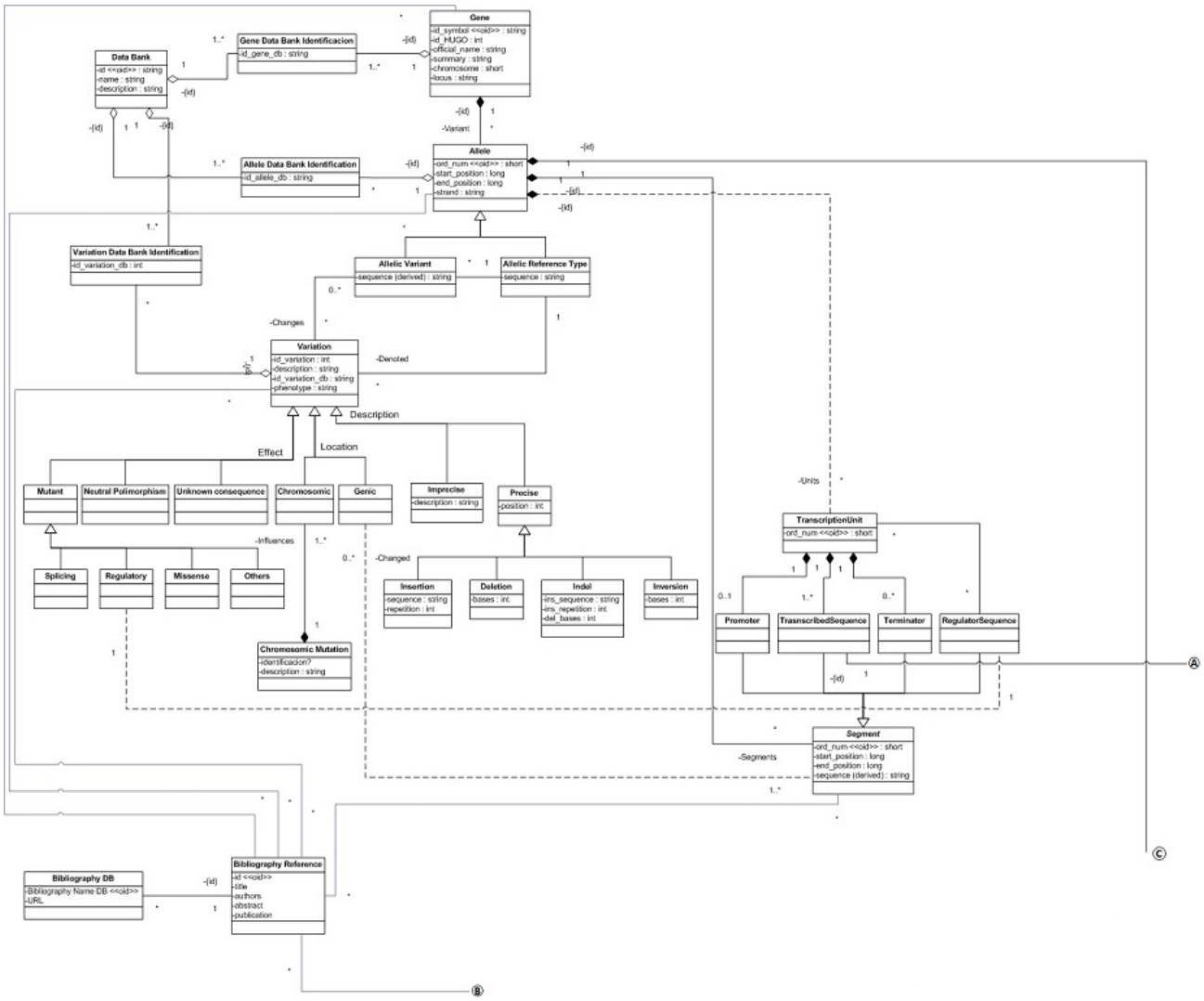


Figura 4.21: Vista Gene-Mutation

Así que aplicando las reglas de transformación definidas, el esquema relacional de la vista Gene-Mutation queda de la siguiente forma:

- ***Gene***(id_symbol:string, id_HUGO:integer, official_name:string, summary:string, chromosome:integer, locus:string)
  CP: {id_symbol}
  VNN: {official_name, chromosome, locus, id_HUGO}
  Único: {id_HUGO}

- ***Allele***(id_gene:string, allele_num:integer, start_position:integer, end_position:integer, strand:char)
  CP: {id_gene, allele_num}
  CAj:{id_gene}→***Gene***(id_gene)

- ***DataBank***(id_data_bank:string, name:string, description:string)
  CP: {id_data_bank}
  VNN: {name}

- ***Gene_ DataBank_ Ident***(id_gene:string, id_data_bank:string, id_gene_db:string)
  CP: {id_ gene, id_data_bank}
  VNN: {id_gene_db}
  CAj: {id _gene}→***Gene***(id _symbol) Borrado en cascada
  CAj:{id_data_bank}→***DataBank***(id_data_bank) Borrado en cascada
  RInt: Todo Gene y todo DataBank aparece en Gene_DataBank_Ident

- ***Allele_ DataBank_ Ident***(id_gene:string, allele_num:integer, id_data_bank:string, id_allele_db:string)
  CP:{id_gene, allele_num, id_data_bank}
  VNN:{id_allele_db}
  CAj:{id_gene, allele_num}→***Allele***(id_gene, allele_num) Borrado en cascada
  CAj:{id_data_bank}→***DataBank***(id_data_bank)Borrado en cascada
  RInt: Todo Allele y todo DataBank aparecen en Allele_DataBank_Ident

- ***Allele_ Reference_ Type***(id_gene:string, allele_num:int, sequence:string)
  CP:{id_gene, allele_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

- ***Allele_ Variant***(id_gene:string, allele_num:int, allele_num_RT:int, sequence:string)
  CP: {id_gene, allele_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

CAj: {id_gene_RT, allele_num_RT}→***Allele_ Reference_ Type***(id_gene, allele_num)

- ***Variation***(id_variation:int, id_gene_RT:string, allele_num_RT:int, phenotype:string, id_data_bank:string, id_variation_db:string, description:string)
  CP:{id_variation}
  CAj: {id_data_bank}→***DataBank***(id_data_bank)
  CAj: {id_gene_RT, allele_num_RT}→***Allele_ Reference_ Type***(id_gene, allele_num)

- ***Changes***(id_gene:string, allele_num:int, id_variation:int)
  CP: {id_gene, allele_num,id_variation}
  CAj: {id_variation}→***Variation***(id_variation)
  CAj: {id_gene, allele_num}→***Allele_ Variant***(id_gene, allele_num)

- ***Mutant***(id_variation:int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Neutral_ Polimorphism***(id_variation:int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Unknown_ Consequence***(id_variation:int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Chromosomic***(id_variation:int, id_chromosomic_mutation:string)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)
  VNN: {id_chromosomic_mutation}
  CAj: {id_chromosomic_mutation}
  → ***Chromosomic_ Mutation***(id_chromosomic_mutation)

- ***Genic***(id_variation: int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Imprecise***(id_variation:int, description:string, type:string)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Precise***(id_variation: int, position: int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)
  VNN: {position}

- ***Splicing***(id_variation:int)
  CP: {id_variation}
  CAj: {id_variation}→***Mutant***(id_variation)

- ***Regulatory***(id_variation: int)
  CP: {id_variation}
  CAj: {id_variation}→***Mutant***(id_variation)

- ***Missense***(id_variation: int)
  CP: {id_variation}
  CAj: {id_variation}→***Mutant***(id_variation)

- ***Others***(id_variation: int)
  CP: {id_variation}
  CAj: {id_variation}→***Mutant***(id_variation)

- ***Chromosomic_Mutation***(id_chromosomic_mutation:string, description:string)
  CP: {id_chromosomic_mutation}
  RInt: Todo Chromosomic_Mutation aparece en Chromosomic

- ***Insertion***(id_variation:int, sequence:string, repetiton:int)
  CP: {id_variation}

CAj: {id_variation}→***Precise***(id_variation)
VNN: {sequence, repetition}

- ***Deletion***(id_variation: int, bases: int)
  CP: {id_variation}
  CAj: {id_variation}→***Precise***(id_variation)
  VNN: {bases}

- ***Indel***(id_variation:int, ins_sequence:string, ins_repetiton:int, bases:int)
  CP: {id_variation}
  CAj: {id_variation}→***Precise***(id_variation)
  VNN: {sequence, repetition, bases}

- ***Inversion***(id_variation:int, bases:int)
  CP: {id_variation}
  CAj: {id_variation}→***Precise***(id_variation)
  VNN: {bases}

- ***Segment***(id_gene:string, allele_num:int, segment_num:int, star_position:int, end_position:int, sequence:string)
  CP: {id_gene, allele_num, segment_num }
  VNN: {star_position, end_position, sequence}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

- ***Transcription_ Unit***(id_gene:string, allele_num:int, trans_unit_num:int)
  CP: {id_gene, allele_num, trans_unit_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

- ***Promoter***(id_gene_S:string, allele_num_S:int, segment_num:int, trans_unit_num:int)
  CP: {id_gene_S, allele_num_S, segment_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene, allele_num, segment_num)
  CAj: {id_gene_S, allele_num_S, trans_unit_num}
  →***Transcription_ Unit***(id_gene, allele_num, trans_unit_num)

VNN: {trans_unit_num}
Único: {id_gene_S, allele_num_S, trans_unit_num}

- ***Trancribed_ Sequence***(id_gene_S:string, allele_num_S:int, segment_num:int, trans_unit_num:int)
  CP: {id_gene_S, allele_num_S, segment_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene, allele_num, segment_num)
  CAj: {id_gene_S, allele_num_S, otrans_unit_num}
  →***Transcription_ Unit***(id_gene, allele_num, trans_unit_num)
  VNN: {trans_unit_num}
  RInt: Todo Transcription_Unit aparece en Transcribed_Sequence.

- ***Terminator***(id_gene_S:string, allele_num_S:int, segment_num:int, trans_unit_num:int)
  CP: {id_gene_S, allele_num_S, segment_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene, allele_num, segment_num)
  CAj: {id_gene_S, allele_num_S, trans_unit_num}
  →***Transcription_ Unit***(id_gene, allele_num, trans_unit_num)
  VNN: {trans_unit_num}

- ***Regulator_ Sequence***(id_gene_S:string, allele_num_S:int, segment_num:int)
  CP: {id_gene_S, allele_num_S, segment_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene, allele_num, segment_num)

- ***Regulates***(id_gene_S:string, allele_num_S:int, segment_num:int, id_gene_TU:string, allele_num_TU:int, trans_unit_num:int)
  CP: {id_gene_S, allele_num_S, segment_num, id_gene_TU, allele_num_TU, trans_unit_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene, allele_num, segment_num)
  CAj: {id_gene_TU, allele_num_TU, trans_unit_num}
  →***Transcription_ Unit***(id_gene, allele_num, trans_unit_num)

- ***Bibliography_ Reference***(id_bib_ref:int, title:string, asbtract:string, publication:string, authors:string, date_pub:date)
  CP: {id_bib_ref}
  VNN: {title, publication}

- ***Reference_ Variation***(id_variation:int, id_bib_ref:int)
  CP: {id_variation,id_bib_ref}
  CAj: {id_variation }→***Variation***(id_variation)
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref)

- ***Reference_ Allele***(id_gene, allele_num:int, id_bib_ref:int)
  CP: {id_gene, allele_num,id_bib_ref}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)
  CAj: {id_bib_ref }→***Bibliography_ Reference***(id_bib_ref)

- ***Reference_ Gene***(id_gene:int, id_bib_ref:int)
  CP:{id_gene, id_bib_ref}
  CAj: {id_gene}→***Gene***(id_gene)
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref)

- ***Reference_ Segment***(id_gene:int, allele_num:int, segment_num:int, id_bib_ref:int)
  CP: {id_gene, allele_num, segment_num, id_bib_ref}
  CAj: {id_gene, allele_num, segment_num, id_bib_ref}→***Segment***(id_gene, allele_num, segment_num) Borrado en cascada
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- ***Bibliography_ DataBank***(bid_databank_name:string, id_bib_ref:int, URL:string)
  CP: {bid_databank_name, id_bib_ref }
  Caj: {id_bib_ref} →***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- ***Reference_ Spliced_ Transcript***(id_gene:int, allele_num:int, spliced_transcript_num:int, id_bib_ref:int)

CP: {id_gene, allele_num, spliced_transcript_num, id_bib_ref}
CAj: {id_gene, allele_num, spliced_transcript_num}
→***Spliced_ Transcript***(id_gene, allele_num, spliced_transcript_num) Borrado en cascada
CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

Así se termina la traducción de la vista Gene_Mutation del Esquema Conceptual al Esquema Relacional. A continuación se muestra el Esquema Relacional resultante de las transformaciones aplicadas (Ver Fig. 4.22).

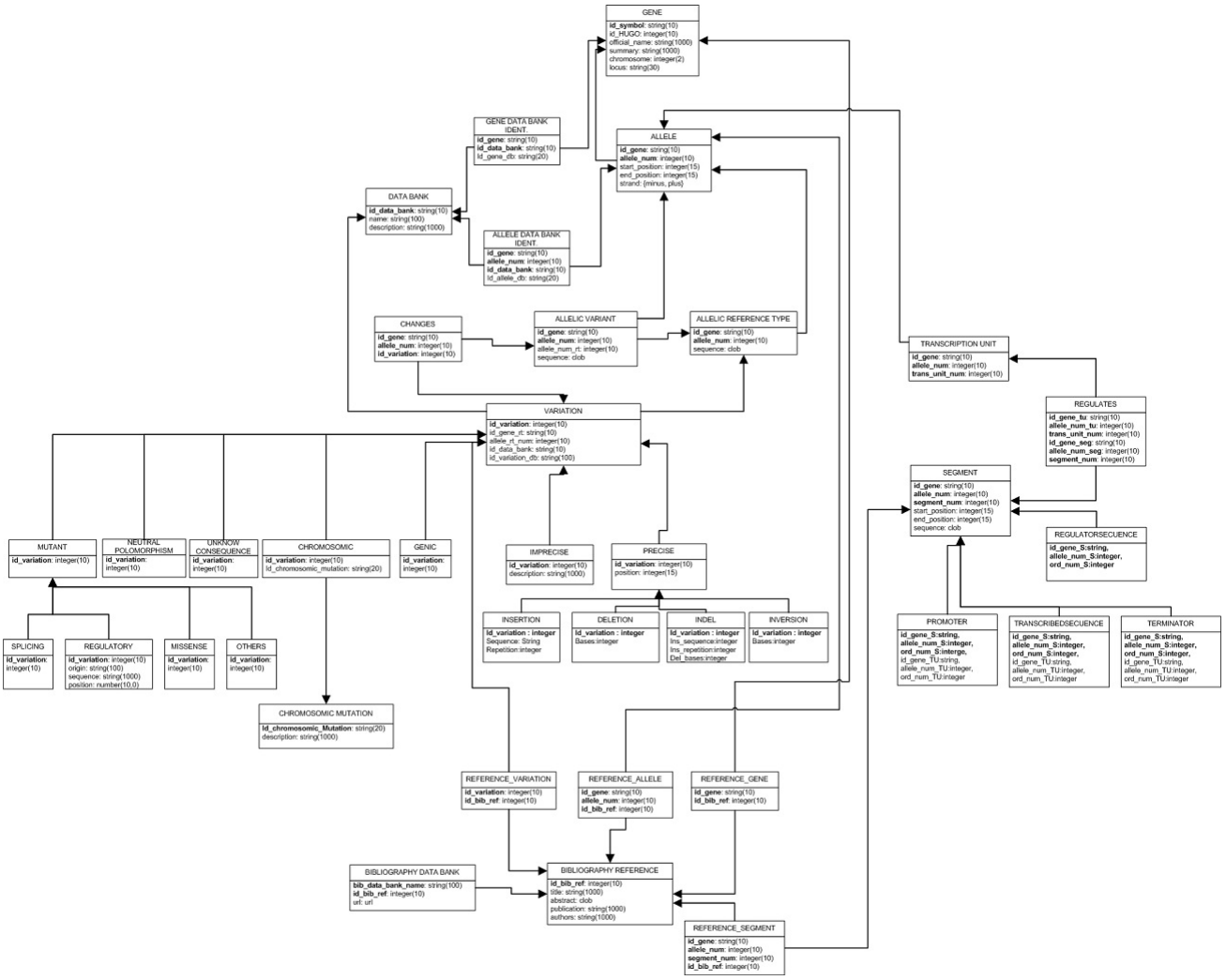


Figura 4.22: Esquema Relacional Vista Gene-Mutation

Para la vista de Transcripción (Transcription View) del ECGH (ver Fig.4.23), el Esquema Relacional resultante de las transformaciones es el siguiente:

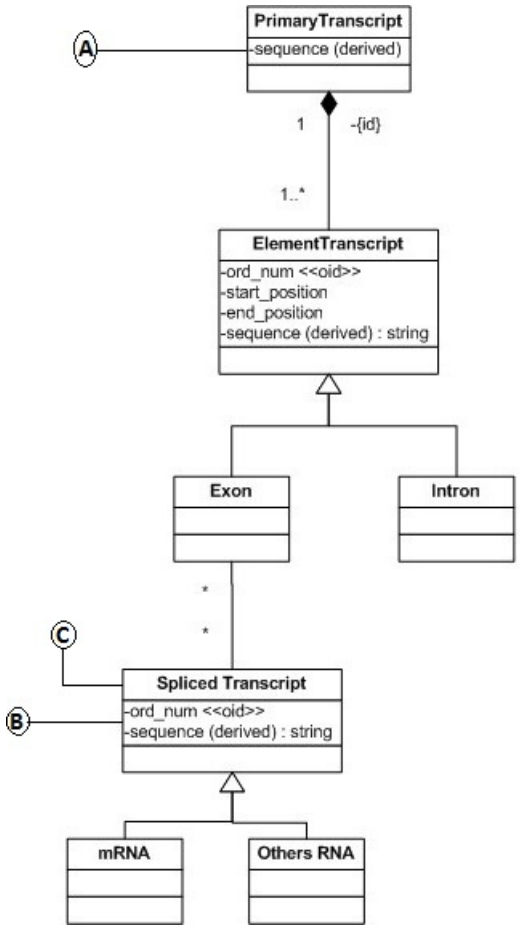


Figura 4.23: Vista Transcription del ECGH

- ***Primary_ Transcrip***(id_gene:int, allele_num:int, segment_num:int, sequence:string)
  CP: {id_gene, allele_num, segment_num}
  CAj: {id_gene, allele_num, segment_num}→***Transcribed_ Sequence***(id_gene, allele_num, segment_num)

- **Element_Transcript**(id_gene:int, allele_num:int, segment_num:int, element_num:int, start_position:int, end_position:int, sequence:int)
  CP: {id_gene, allele_num, segment_num, element_num}
  CAj: {id_gene, allele_num, segment_num}→***Primary_ Transcript***(id_gene, allele_num, segment_num)

- ***Exon***(id_gene:int, allele_num:int, segment_num:int, element_num:int)
  CP: {id_gene, allele_num, segment_num, element_num}
  CAj: {id_gene, allele_num, segment_num}→***Element_ Transcript***(id_gene, allele_num, segment_num)

- ***Intron***(id_gene:int, allele_num:int, segment_num:int, element_num:int)
  CP: {id_gene, allele_num, segment_num, element_num}
  CAj: {id_gene, allele_num, segment_num}→***Element_ Transcript***(id_gene, allele_num, segment_num)

- ***Spliced_ Transcript***(id_gene:int, allele_num:int, spliced_transcript_num:int, sequence:string)
  CP: {id_gene, allele_num, spliced_transcript_num}
  CAj: {id_gene,allele_num}→***Allele***(id_gene, allele_num)
  VNN: {sequence}

- ***Produces***(id_gene:int, allele_num:int, segment_num:int, element_num:int, splicing_transcript_num:int)
  CP: {id_gene, allele_num, segment_num, element_num, splicing_transcript_num}
  CAj: {id_gene, allele_num, segment_num, element_num}→***Exon***(id_gene, allele_num, segment_num, element_num)
  CAj: {id_gene,allele_num, splicing_transcript_num}→***Spliced_ Transcript***(id_gene, allele_num, spliced_transcript_num)

- ***mRNA***(id_gene: int, allele_num: int, spliced_transcript_num:int)
  CP: {id_gene, allele_num, spliced_transcript_num}
  CAj: {id_gene, allele_num, spliced_transcript_num}→ ***Spliced_ Transcript***(id_gene, allele_num, spliced_transcript_num)

- **Others_RNA**(id_gene:int, allele_num:int, spliced_transcript_num:int)
  CP: {id_gene, allele_num, spliced_transcript_num}
  CAj: {id_gene, allele_num, spliced_transcript_num}→ ***Spliced_ Transcript***(id_gene, allele_num, spliced_transcript_num)

En la figura 4.19 se puede observar el esquema relacional resultante de la vista Trasncription del esquema conceptual.

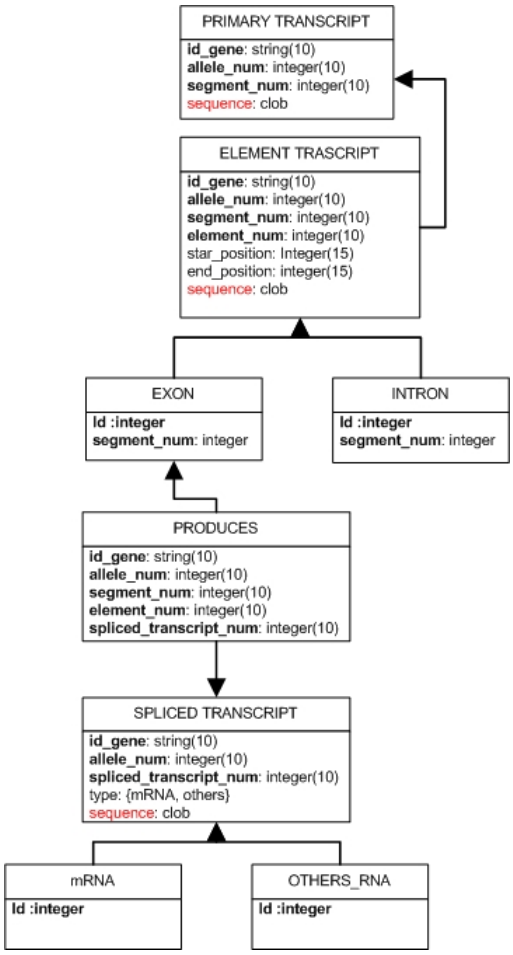


Figura 4.24: Esquema Relacional Vista Transcription

4.3.1. Afinamiento del esquema lógico

En este apartado se va a reconsiderar el esquema lógico obtenido en el apartado anterior con el objetivo de simplificar el esquema y hacerlo más manejable desde las aplicaciones. Para ello, se van a eliminar algunas relaciones asociadas a clases especializadas. Este cambio supondrá introducer en la relación de la clase general un atributo tipo que permita discriminar a qué especialización pertence cada ocurrencia. En ocasiones también será necesario introducir alguna restricción añadida al esquema lógico.

- ***Gene***(id_symbol:string, id_HUGO:integer, official_name:string, summary:string, chromosome:integer, locus:string)
  CP: {id_symbol}
  VNN: {official_name, chromosome, locus}
  Único: {id_HUGO}

- ***Allele***(id_gene:string, allele_num:integer, start_position:integer,end_position:integer, strand:char)
  CP: {id_gene, allele_num}
  CAj: {id_gene}→***Gene*** (id_gene)

- ***DataBank***(id_data_bank:string, name:string, description:string)
  CP: {id_data_bank}
  VNN: {name}

- ***Gene_DataBank_Ident***(id_ gene:string, id_data_bank:string, id_gene_db:string)
  CP: {id_ gene, id_data_bank}
  VNN: {id_gene_db}
  CAj: {id _gene}→***Gene***(id _symbol)
  CAj: {id_data_bank}→***DataBank***(id_data_bank)
  Único: {id_data_bank, id_gene_db}
  RInt: Todo Gene y todo DataBank aparece en Gene_DataBank_Ident

- ***Allele_DataBank_Ident***(id _gene:string, allele_num:integer, id_data_bank:string, id_allele_db:string)
  CP: {id_gene,allele_num, id_data_bank}
  VNN: {id_allele_db}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)
  CAj: {id_data_bank}→***DataBank***(id_data_bank)
  Único: {id_data_bank, id_allele_db}
  RInt: Todo Allele y todo DataBank aparece en Allele_DataBank_Ident

- ***Allelic_Reference_Type***(id_gene:string, allele_num:int, sequence:string)
  CP: {id_gene, allele_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

- ***Allelic_ Variant***(id_gene:string, allele_num:int, id_gene_RT:string, allele_num_RT:int, sequence:string)
  CP: {id_gene, allele_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)
  CAj: {id_gene_RT, allele_num_RT}→***Allelic_ Reference_ Type***(id_gene, allele_num)

En la relación ***Variation*** se introducen varios atributos para indicar a qué especialización pertenece cada variación eliminando por ello algunas de las relaciones asociadas a clases especializadas. Para controlar la integridad de la información almacenada, se introducen algunas restricciones de integridad:

- ***Variation***(id_variation:int, id_gene_RT:string, allele_num_RT:int, specialization_effect:string, specialization_mutant:string, id_data_bank: string, id_variation_db: string, description: string, specialization_localization:string)
  CP: {id_variation}
  CAj: {id_data_bank}→DataBank(id_data_bank)
  CAj: {id_gene_RT, allele_num_RT}→Allelic_Reference_Type (id_gene, allele_num)
  - Restricciones de integridad:
    - Los valores posibles para el atributo specialization_effect son {'M', 'N', U'} para representar las especializaciones Mutant, Neutral_Polomorphism y Unknown_Consequence.
    - Los valores posibles para el atributo specialization_localization son {'C', 'G'} para representar las especializaciones Chromosomic y Genic.
    - Los valores posibles para el atributo specialization_mutant son {'S', 'R', 'M', 'O'} para representar las especializaciones Splicig, Regulatory, Missense y Others.
- ***Changes***(id_gene:string, allele_num:int, id_variation:int)
  CP: {id_gene, allele_num, id_variation}

CAj: {id_variation }→***Variation***(id_variation)
CAj: {id_gene, allele_num}→***Allelic_ Variant***(id_gene, allele_num)

- ***Chromosomic***(id_variation:int, id_chromosomic_mutation:string)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)
  VNN: {id_chromosomic_mutation}
  CAj: {id_chromosomic_mutation}→
  Chromosomic_ Mutation(id_chromosomic_mutation)

- ***Imprecise***(id_variation: int, description: string,type: string)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

En la relación ***Precise*** se introducen varios atributos para indicar a qué especialización pertenece una variación precisa eliminando por ello algunas de las relaciones asociadas a clases especializadas. Para controlar la integridad de la información almacenada, se introducen algunas restricciones de integridad:

- ***Precise***(id_variation:int, position:int, type:string, ins_seq:string, ins_repetition:int, num_bases:int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)
  VNN: {position, type}

  - Restricciones de integridad:
    - Los valores posibles para el atributo type son {'IS', 'DE', 'ID', 'IV'} para representar las especializaciones ***Insertion, Deletion, Indel*** y ***Inversion***.

- ***Regulatory***(id_variation: int)
  CP: {id_variation}
  CAj: {id_variation}→***Variation***(id_variation)

- ***Chromosomic_Mutation***(id_chromosomic_mutation:string, description:string)
  CP: {id_chromosomic_mutation}
  RInt: Todo Chromosomic_Mutation aparece en Chromosomic

En la relación ***Segment*** se introducen un atributo para indicar a qué especialización pertenece un segmento eliminando por ello algunas de las relaciones asociadas a clases especializadas. Para controlar la integridad de la información almacenada, se introducen algunas restricciones de integridad:

- ***Segment***(id_gene:string, allele_num:int, segment_num:int, start_position:int, end_position:int, sequence:string, trans_unit_num:int, type:string)
  CP: {id_gene, allele_num, segment_num }
  VNN: {star_position, end_position, sequence}
  CAj: {id_gene, allele_num}→Allele(id_gene, allele_num)
  CAj: {id_gene, allele_num, trans_unit_num}→Transcription_Uni(id_gene, allele_num, trans_unit_num)

  - Restricciones de integridad:
    - Los valores posibles para el atributo type son {'PR', 'TS', 'TE', 'RS'} para representar las especializaciones ***Promotor***, ***TranscribedSequence***, ***Terminator*** y ***RegulatorSequence***.

- ***Transcription_Uni***(id_gene:string, allele_num:int, trans_unit_num:int)
  CP: {id_gene, allele_num, trans_unit_num}
  CAj: {id_gene, allele_num}→***Allele***(id_gene, allele_num)

- ***Regulates***(id_gene_S:string, allele_num_S:int, segment_num:int, id_gene_TU:string, allele_num_TU:int, trans_unit_num:int)
  CP: {id_gene_S, allele_num_S, segment_num, id_gene_TU, allele_num_TU, trans_unit_num}
  CAj: {id_gene_S, allele_num_S, segment_num}→***Segment***(id_gene,

allele_num, segment_num)
CAj: {id_gene_TU, allele_num_TU, trans_unit_num}→ ***Transcription_ Unit*** (id_gene, allele_num, trans_unit_num)

- ***Bibliography_ Reference***(id_bib_ref:int, title:string, asbtract:string, publication:string, authors:string, date_pub:date)
  CP: {id_bib_ref}
  VNN: {title, publication}

- **Reference_ Variation**(id_variation:int, id_bib_ref:int)
  CP: {id_variation, id_bib_ref}
  CAj: {id_variation }→***Variation***(id_variation)
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref)

- ***Reference_ Allele***(id_gene:int, allele_num:int, id_bib_ref:int)
  CP: {id_gene, allele_num, id_bib_ref}
  CAj: {id_gene, allele_num}→***Allele***(id_gene,allele_num) Borrado en cascada
  CAj: {id_bib_ref }→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- ***Reference_ Gene***(id_gene:int, id_bib_ref:int)
  CP: {id_gene,id_bib_ref}
  CAj: {id_gene}→***Gene***(id_gene) Borrado en cascada
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- ***Reference_ Segment***(id_gene:int, allele_num:int, segment_num:int, id_bib_ref:int)
  CP: {id_gene, allele_num, segment_num, id_bib_ref}
  CAj: {id_gene, allele_num, segment_num}→***Segment***(id_gene, allele_num, segment_num) Borrado en cascada
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- ***Bibliography_ DataBank***(bid_databank_name:string, id_bib_ref int, URL:string)
  CP: { bid_databank_name, id_bib_ref }
  Caj: {id_bib_ref} →***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

- **Reference_Spliced_Transcript**(id_gene:int, allele_num:int, spliced_transcript_num:int, id_bib_ref:int)
  CP: {id_gene, allele_num, spliced_transcript_num, id_bib_ref}
  CAj: { id_gene, allele_num, spliced_transcript_num }→***Spliced_ Transcript***(id_gene, allele_num, spliced_transcript_num) Borrado en cascada
  CAj: {id_bib_ref}→***Bibliography_ Reference***(id_bib_ref) Borrado en cascada

Para la vista de Transcripción el refinamiento del esquema queda de la siguiente forma:

- ***Primary_ Transcript***(id_gene:int, allele_num:int, segment_num:int, sequence:string)
  CP: {id_gene, allele_num, segment_num}
  CAj: {id_gene, allele_num, segment_num}→***Segment***(id_gene, allele_num, segment_num)

En la relación ***Element_ Transcript*** se introducen un atributo para indicar a qué especialización pertenece un elemento transcrito eliminando por ello algunas de las relaciones asociadas a clases especializadas. Para controlar la integridad de la información almacenada, se introducen algunas restricciones de integridad:

- ***Element_ Transcript***(id_gene:int, allele_num:int, segment_num:int, element_num:int, start_position:int, end_position:int, sequence:int, element_type:string)
  CP: {id_gene, allele_num, segment_num, element_num}

CAj: {id_gene, allele_num, segment_num}→***Primary_ Transcript***(id_gene, allele_num, segment_num)

- Restricciones de integridad:
  - Los valores posibles para el atributo element_type son {'E', 'I'} para representar las especializaciones ***Exon*** e ***Intron***.

En la relación ***Spliced_ Transcript*** se introducen un atributo para indicar a qué especialización pertenece un tránscrito eliminando por ello algunas de las relaciones asociadas a clases especializadas. Para controlar la integridad de la información almacenada, se introducen algunas restricciones de integridad:

- ***Spliced_ Transcript***(id_gene:int, allele_num:int, spliced_transcript_num:int, sequence:string, type:string)
  CP: {id_gene, allele_num, spliced_transcript_num}
  CAj: {id_gene,allele_num}→***Allele***(id_gene,allele_num)
  VNN: {sequence}

  - Restricciones de integridad:
    - Los valores posibles para el atributo type son {'mRNA', 'others'} para representar las especializaciones en ***mRNA*** y ***Others***.

- ***Produces***(id_gene:int, allele_num:int, segment_num:int, element_num:int, splicing_transcript_num:int)
  CP: {id_gene, allele_num, segment_num, element_num, splicing_transcript_num}
  CAj: {id_gene,allele_num, segment_num, element_num}
  →***Element_ Transcript***(id_gene, allele_num, segment_num, element_num)
  CAj: {id_gene,allele_num, splicing_transcript_num }
  → ***Spliced_ Transcript***(id_gene, allele_num, splicing_transcript_num)

4.4. Diseño físico e implantación de la Base de Datos del Genoma Humano

En este apartado se presenta la definición de la base de datos para el SGBD que se va a utilizar que es el Oracle 10g. Para cada relación se han definido además de todos los atributos y todas las restricciones definidas los tamaños asignados para la gestión de la información y algunos índices cuando se ha considerado necesario. Las instrucciones de creación del esquema de la base de datos son las siguientes:

- CREATE TABLE "GENE" (
  "ID_SYMBOL" VARCHAR2(10 BYTE) NOT NULL ENABLE,
  "ID_HUGO" NUMBER(10,0) NOT NULL ENABLE,
  "OFFICIAL_NAME" VARCHAR2(1000 BYTE) NOT NULL ENABLE,
  "SUMMARY" VARCHAR2(1000 BYTE),
  "CHROMOSOME" NUMBER(2,0) NOT NULL ENABLE,
  "LOCUS" VARCHAR2(30 BYTE) NOT NULL ENABLE,
  CONSTRAINT "GENE_PK" PRIMARY KEY ("ID_SYMBOL")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "USERS" ENABLE,
  CONSTRAINT "GENE_UK1" UNIQUE ("ID_HUGO")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
  COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
  MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "USERS" ENABLE,

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;
CREATE UNIQUE INDEX "GENE_PK" ON "GENE" ("ID_SYMBOL")
PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;
CREATE UNIQUE INDEX "GENE_UK1" ON "GENE" ("ID_HUGO")
PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "ALLELE" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"START_POSITION" NUMBER(15,0),
"END_POSITION" NUMBER(15,0),
"STRAND" CHAR(1 BYTE),
CONSTRAINT "ALLELE_CHK1" CHECK (strand in ('M','P')) ENABLE,
CONSTRAINT "ALLELE_PK" PRIMARY KEY ("ID_GENE", "ALLE-

LE_NUM") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "USERS" ENABLE, CONSTRAINT "ALLELE_GENE_FK1" FOREIGN KEY ("ID_GENE") REFERENCES "GENE" ("ID_SYMBOL") ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "USERS" ;

- CREATE TABLE "DATABANK" (
"ID_DATABANK" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"DESCRIPTION" VARCHAR2(1000 BYTE),
CONSTRAINT "DATA_BANK_PK" PRIMARY KEY ("ID_DATABANK")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "GENE_DATABANK_IDENT" (

"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ID_DATABANK" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ID_GENE_DB" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "GENE_DATABANK_IDENT_PK" PRIMARY KEY
("ID_GENE", "ID_DATABANK") USING INDEX PCTFREE 10 INITRANS
2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "GENE_DB_IDENT_DATABANK_FK1" FOREIGN
KEY ("ID_DATABANK") REFERENCES "DATABANK" ("ID_DATABANK")
ON DELETE CASCADE ENABLE,
CONSTRAINT "GENE_DATABANK_IDENT_GENE_FK1" FOREIGN
KEY ("ID_GENE") REFERENCES "GENE" ("ID_SYMBOL") ON
DELETE CASCADE ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NO-
COMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "ALLELE_DATABANK_IDENT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_DATABANK" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ID_ALLELE_DB" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT
"ALLELE_DATABANK_IDENT_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "ID_DATABANK") USING INDEX PCTFREE

10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "ALLELE_DATABANK_IDENT_DAT_FK1" FO-
REIGN KEY ("ID_DATABANK") REFERENCES "DATABANK"
("ID_DATABANK") ON DELETE CASCADE ENABLE,
CONSTRAINT "ALLELE_DB_IDENT_ALLELE_FK1" FOREIGN
KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELE"
("ID_GENE","ALLELE_NUM") ON DELETE CASCADE ENABLE
)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NO-
COMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "ALLELIC_REFERENCE_TYPE" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEQUENCE" CLOB,
CONSTRAINT "REFERENCE_TYPE_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM") USING INDEX PCTFREE 10 INITRANS 2 MAX-
TRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,

CONSTRAINT "REFERENCE_TYPE_ALLELE_FK1" FOREIGN KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELE" ("ID_GENE", "ALLELE_NUM") ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" LOB ("SEQUENCE")
(TABLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK 8192 PCTVERSION 10 NOCACHE LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)) ;

- CREATE TABLE "ALLELIC_VARIANT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ALLELE_NUM_RT" NUMBER(10,0) NOT NULL ENABLE,
"SEQUENCE" CLOB,
CONSTRAINT "ALLELIC_VARIANT_PK" PRIMARY KEY ("ID_GENE", "ALLELE_NUM") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "ALLELIC_VARIANT_REFERENCE_FK1" FOREIGN KEY ("ID_GENE", "ALLELE_NUM_RT") REFERENCES "ALLELIC_REFERENCE_TYPE" ("ID_GENE", "ALLELE_NUM") ENABLE,
CONSTRAINT "ALLELIC_VARIANT_ALLELE_FK1" FOREIGN

KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELE"
("ID_GENE", "ALLELE_NUM") ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NO-
COMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" LOB ("SEQUENCE") STORE AS (TA-
BLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK 8192
PCTVERSION 10 NOCACHE
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREE-
LIST GROUPS 1 BUFFER_POOL DEFAULT)) ;

- CREATE TABLE "VARIATION" (
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
"ID_GENE_RT" VARCHAR2(10 BYTE),
"ID_ALLELE_NUM_RT" NUMBER(10,0),
"SPECIALIZATION_EFFECT" CHAR(1 BYTE),
"SPECIALIZATION_MUTANT" CHAR(1 BYTE),
"SPECIALIZATION_LOCALIZATION" CHAR(1 BYTE),
"ID_DATA_BANK" VARCHAR2(10 BYTE),
"ID_VARIATION_DB" VARCHAR2(100 BYTE),
"DESCRIPTION" VARCHAR2(1000 BYTE),
CONSTRAINT "VARIATION_PK" PRIMARY KEY ("ID_VARIATION")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE, CONSTRAINT "VARIATION_CHK1"
CHECK (SPECIALIZATION_EFFECT IN ('M','N','U')) ENABLE,

CONSTRAINT "VARIATION_CHK2" CHECK (SPECIALIZATION_LOCALIZATION
IN ('C','G')) ENABLE,
CONSTRAINT "VARIATION_CHK3" CHECK (SPECIALIZATION_MUTANT
IN ('S','R','M','O')) ENABLE,
CONSTRAINT "VARIATION_REFERENCE_TYPE_FK1" FOREIGN
KEY ("ID_GENE_RT", "ID_ALLELE_NUM_RT") REFERENCES
"ALLELIC_REFERENCE_TYPE" ("ID_GENE", "ALLELE_NUM")
ENABLE,
CONSTRAINT "VARIATION_DATABANK_FK1" FOREIGN KEY
("ID_DATA_BANK") REFERENCES "DATABANK" ("ID_DATABANK")
ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NO-
COMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "CHANGES" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "CHANGES_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "ID_VARIATION") ENABLE,
CONSTRAINT "CHANGES_ALLELIC_VARIANT_FK1" FOREIGN
KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELIC_VARIANT"
("ID_GENE", "ALLELE_NUM") ENABLE,
CONSTRAINT "CHANGES_VARIATION_FK1" FOREIGN KEY ("ID_VARIATION")
REFERENCES "VARIATION" ("ID_VARIATION") ENABLE)
ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS
2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS

1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS"
PCTTHRESHOLD 50;

- CREATE TABLE "CHROMOSOMIC" (
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
"ID_CHROMOSOMIC_MUTATION" VARCHAR2(20 BYTE) NOT NULL ENABLE,
CONSTRAINT "CHROMOSOMIC_PK" PRIMARY KEY ("ID_VARIATION")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "CHROMOSOMIC_VARIATION_FK1" FOREIGN KEY ("ID_VARIATION") REFERENCES "VARIATION" ("ID_VARIATION") ENABLE,
CONSTRAINT "CHROMOSOMIC_CHROMOSOMIC_M_FK1" FOREIGN KEY ("ID_CHROMOSOMIC_MUTATION") REFERENCES "CHROMOSOMIC_MUTATION" ("ID") ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "IMPRECISE" (
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
"DESCRIPTION" VARCHAR2(1000 BYTE),

CONSTRAINT "IMPRECISE_PK" PRIMARY KEY ("ID_VARIATION")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "IMPRECISE_VARIATION_FK1" FOREIGN KEY
("ID_VARIATION") REFERENCES "VARIATION" ("ID_VARIATION")
ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NO-
COMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "PRECISE" (
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
"POSITION" NUMBER(15,0) NOT NULL ENABLE,
"TYPE" CHAR(2 BYTE) NOT NULL ENABLE,
"INS_SEQUENCE" CLOB,
"INS_REPETITION" NUMBER(10,0),
"NUM_BASES" NUMBER(10,0),
CONSTRAINT "PRECISE_PK" PRIMARY KEY ("ID_VARIATION")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "PRECISE_CHK1" CHECK (TYPE IN ('IS','DE','ID','IV'))

ENABLE,
CONSTRAINT "PRECISE_VARIATION_FK1" FOREIGN KEY ("ID_VARIATION")
REFERENCES "VARIATION" ("ID_VARIATION") ENABLE)
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" LOB ("INS_SEQUENCE") STORE AS (
TABLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK
8192 PCTVERSION 10 NOCACHE
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)) ;

- CREATE TABLE "REGULATORY" (
"ID_VARIATION" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ORIGIN" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"SEQUENCE" VARCHAR2(1000 BYTE) NOT NULL ENABLE, "POSITION" NUMBER(10,0) NOT NULL ENABLE, CONSTRAINT "REGULATORY_PK" PRIMARY KEY ("ID_VARIATION") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "REGULATORY_VARIATION_FK1" FOREIGN KEY
("POSITION") REFERENCES "VARIATION" ("ID_VARIATION")
ENABLE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS
255 NOCOMPRESS

LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "CHROMOSOMIC_MUTATION" (
"ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,
"DESCRIPTION" VARCHAR2(1000 BYTE) NOT NULL ENABLE,
CONSTRAINT
"CHROMOSOMIC_MUTATION_PK" PRIMARY KEY ("ID") USING
INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE) PCTFREE 10 PCTUSED 40
INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "SEGMENT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"START_POSITION" NUMBER(15,0) NOT NULL ENABLE,
"END_POSITION" NUMBER(15,0) NOT NULL ENABLE,
"SEQUENCE" CLOB NOT NULL ENABLE,
"TRANS_UNIT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"TYPE" CHAR(2 BYTE) NOT NULL ENABLE,
CONSTRAINT "SEGMENT_TYPE_CHK1" CHECK (type in ('PR','TS','TE','RS')

) ENABLE,
CONSTRAINT "SEGMENT_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "SEGMENT_NUM") USING INDEX PCTFREE
10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "SEGMENT_TRANSCRIPTION_UNI_FK1" FOREIGN
KEY ("ID_GENE", "ALLELE_NUM", "TRANS_UNIT_NUM") RE-
FERENCES "TRANSCRIPTION_UNI" ("ID_GENE", "ALLELE_NUM",
"TRANS_UNIT_NUM") ENABLE) PCTFREE 10 PCTUSED 40
INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" LOB ("SEQUENCE") STORE AS (TA-
BLESPACE "USERS" ENABLE STORAGE IN ROW CHUNK 8192
PCTVERSION 10 NOCACHE
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)) ;

- CREATE TABLE "TRANSCRIPTION_UNI" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"TRANS_UNIT_NUM" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "TRANSCRIPTION_UNI_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "TRANS_UNIT_NUM") ENABLE) ORGANI-
ZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS 2 MAX-

TRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "REGULATES" (
"ID_GENE_TU" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM_TU" NUMBER(10,0) NOT NULL ENABLE,
"TRANS_UNIT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_GENE_SEG" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM_SET" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REGULATES_PK" PRIMARY KEY ("ID_GENE_TU",
"ALLELE_NUM_TU", "TRANS_UNIT_NUM", "ID_GENE_SEG",
"ALLELE_NUM_SET", "SEGMENT_NUM") ENABLE,
CONSTRAINT "REGULATES_TRANSCRIPTION_U_FK1" FOREIGN
KEY ("ID_GENE_TU", "ALLELE_NUM_TU", "TRANS_UNIT_NUM")
REFERENCES "TRANSCRIPTION_UNI" ("ID_GENE", "ALLE-
LE_NUM", "TRANS_UNIT_NUM") ENABLE,
CONSTRAINT "REGULATES_SEGMENT_FK1" FOREIGN KEY
("ID_GENE_TU", "ALLELE_NUM_TU", "SEGMENT_NUM") RE-
FERENCES "SEGMENT" ("ID_GENE", "ALLELE_NUM", "SEG-
MENT_NUM") ON DELETE CASCADE ENABLE)
ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS
2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "BIBLIOGRAPHY_REFERENCE" (

"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
"TITLE" VARCHAR2(1000 BYTE) NOT NULL ENABLE,
"ABSTRACT" VARCHAR2(2000 BYTE),
"PUBLICATION" VARCHAR2(1000 BYTE) NOT NULL ENABLE,
"AUTHORS" VARCHAR2(1000 BYTE),
CONSTRAINT "BIBLIOGRAPHY_REFERENCE_PK" PRIMARY
KEY ("ID_BIB_REF") USING INDEX PCTFREE 10 INITRANS 2
MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE) PCTFREE 10 PCTUSED 40
INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "REFERENCE_VARIATION" (
"ID_VARIATION" NUMBER(10,0) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REFERENCE_VARIATION_PK" PRIMARY KEY
("ID_VARIATION", "ID_BIB_REF") ENABLE,
CONSTRAINT "REFERENCE_VARIATION_VARIA_FK1" FOREIGN
KEY ("ID_VARIATION") REFERENCES "VARIATION" ("ID_VARIATION")
ENABLE,
CONSTRAINT "REFERENCE_VARIATION_BIBLI_FK1" FOREIGN
KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE"
("ID_BIB_REF") ENABLE) ORGANIZATION INDEX NOCOM-
PRESS PCTFREE 10 INITRANS 2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS

1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "REFERENCE_ALLELE" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REFERENCE_ALLELE_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "ID_BIB_REF") ENABLE,
CONSTRAINT "REFERENCE_ALLELE_ALLELE_FK1" FOREIGN
KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELE"
("ID_GENE", "ALLELE_NUM") ON DELETE CASCADE ENABLE,
CONSTRAINT "REFERENCE_ALLELE_BIBLIOGR_FK1" FOREIGN
KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE"
("ID_BIB_REF") ON DELETE CASCADE ENABLE)
ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS
2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "REFERENCE_GENE" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REFERENCE_GENE_PK" PRIMARY KEY ("ID_GENE",
"ID_BIB_REF") ENABLE,
CONSTRAINT "REFERENCE_GENE_GENE_FK1" FOREIGN KEY
("ID_GENE") REFERENCES "GENE" ("ID_SYMBOL") ON DE-
LETE CASCADE ENABLE,
CONSTRAINT "REFERENCE_GENE_BIBLIOGRAP_FK1" FOREIGN

KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE"
("ID_BIB_REF") ON DELETE CASCADE ENABLE)
ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS
2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "REFERENCE_SEGMENT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REFERENCE_SEGMENT_PK" PRIMARY KEY
("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM", "ID_BIB_REF")
ENABLE,
CONSTRAINT "REFERENCE_SEGMENT_SEGMENT_FK1" FO-
REIGN KEY ("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM")
REFERENCES "SEGMENT" ("ID_GENE", "ALLELE_NUM", "SEG-
MENT_NUM") ON DELETE CASCADE ENABLE,
CONSTRAINT "REFERENCE_SEGMENT_BIBLIOG_FK1" FOREIGN
KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE"
("ID_BIB_REF") ON DELETE CASCADE ENABLE)
ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS
2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "BIBLIGRAPHY_DATABANK" (

"BID_DATABANK_NAME" VARCHAR2(100 BYTE) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
"URL" VARCHAR2(1000 BYTE),
CONSTRAINT "BIBLIGRAPHY_DATABANK_PK" PRIMARY KEY ("BID_DATABANK_NAME", "ID_BIB_REF") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "BIBLIGRAPHY_DATABANK_BIBL_FK1" FOREIGN KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE" ("ID_BIB_REF") ON DELETE CASCADE ENABLE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "REFERENCE_SPLICED_TRANSCRIPT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SPLICED_TRANSCRIPT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ID_BIB_REF" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "REFERENCE_SPLICED_TRANSCR_PK" PRIMARY KEY ("ID_GENE", "ALLELE_NUM", "SPLICED_TRANSCRIPT_NUM", "ID_BIB_REF") ENABLE,
CONSTRAINT "REF_SPLICED_TRAN_SPLTRAN_FK1" FOREIGN KEY ("ID_GENE", "ALLELE_NUM", "SPLICED_TRANSCRIPT_NUM") REFERENCES "SPLICED_TRANSCRIPT" ("ID_GENE", "ALLE-

LE_NUM", "SPLICED_TRANSCRIPT_NUM") ON DELETE CASCADE ENABLE,
CONSTRAINT "REF_SPLICED_TRAN_BR_FK2" FOREIGN KEY ("ID_BIB_REF") REFERENCES "BIBLIOGRAPHY_REFERENCE" ("ID_BIB_REF") ON DELETE CASCADE ENABLE) ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS 2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "PRIMARY_TRANSCRIPT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "PRIMARY_TRANSCRIPT_PK" PRIMARY KEY ("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM") ENABLE,
CONSTRAINT "PRIMARY_TRANSCRIPT_SEGMEN_FK1" FOREIGN KEY ("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM") REFERENCES "SEGMENT" ("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM") ENABLE) ORGANIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS 2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "PRIMARY_TRANSCRIPT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,

CONSTRAINT "PRIMARY_TRANSCRIPT_PK" PRIMARY KEY
("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM") ENABLE,
CONSTRAINT "PRIMARY_TRANSCRIPT_SEGMEN_FK1" FO-
REIGN KEY ("ID_GENE", "ALLELE_NUM", "SEGMENT_NUM")
REFERENCES "SEGMENT" ("ID_GENE", "ALLELE_NUM", "SEG-
MENT_NUM") ENABLE) ORGANIZATION INDEX NOCOMPRESS
PCTFREE 10 INITRANS 2 MAXTRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

- CREATE TABLE "ELEMENT_TRANSCRIPT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ELEMENT_TYPE" CHAR(1 BYTE) NOT NULL ENABLE,
"ELEMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"START_POSITION" NUMBER(15,0),
"END_POSITION" NUMBER(15,0),
CONSTRAINT "ELEMENT_TRANSCRIPT_PK" PRIMARY KEY
("ID_GENE","ALLELE_NUM","SEGMENT_NUM","ELEMENT_NUM")
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREE-
LISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "ELEMENT_TRANSCRIPT_CHK1" CHECK (ELE-
MENT_TYPE IN ('E','I')) ENABLE,
CONSTRAINT "ELEMENT_TRANSCRIPT_PRIMAR_FK1" FO-
REIGN KEY ("ID_GENE","ALLELE_NUM","SEGMENT_NUM")
REFERENCES "PRIMARY_TRANSCRIPT" ("ID_GENE", "ALLE-

LE_NUM", "SEGMENT_NUM") ENABLE) PCTFREE 10 PCTU-
SED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "SPLICED_TRANSCRIPT" (
"ID_GENE" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SPLICED_TRANSCRIPT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"TYPE" VARCHAR2(20 BYTE),
CONSTRAINT "SPLICED_TRANSCRIPT_TYPE_CHK1" CHECK
(type in ('mRNA','others')) ENABLE,
CONSTRAINT "SPLICED_TRANSCRIPT_PK" PRIMARY KEY ("ID_GENE",
"ALLELE_NUM", "SPLICED_TRANSCRIPT_NUM") USING IN-
DEX PCTFREE 10 INITRANS 2 MAXTRANS 255
COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576
MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ENABLE,
CONSTRAINT "SPLICED_TRANSCRIPT_ALLELE_FK1" FOREIGN
KEY ("ID_GENE", "ALLELE_NUM") REFERENCES "ALLELE"
("ID_GENE", "ALLELE_NUM") ENABLE) PCTFREE 10 PCTU-
SED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" ;

- CREATE TABLE "PRODUCES" (

"ID_GEN" VARCHAR2(10 BYTE) NOT NULL ENABLE,
"ALLELE_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SEGMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"ELEMENT_NUM" NUMBER(10,0) NOT NULL ENABLE,
"SPLICED_TRANSCRIPT_NUM" NUMBER(10,0) NOT NULL ENABLE,
CONSTRAINT "PRODUCES_PK" PRIMARY KEY ("ID_GEN",
"ALLELE_NUM","SEGMENT_NUM","ELEMENT_NUM","SPLI-
CED_TRANSCRIPT_NUM") ENABLE,
CONSTRAINT "PRODUCES_ELEMENT_TRANSCRI_FK1" FO-
REIGN KEY ("ID_GEN", "ALLELE_NUM", "SEGMENT_NUM",
"ELEMENT_NUM") REFERENCES "ELEMENT_TRANSCRIPT"
("ID_GENE","ALLELE_NUM","SEGMENT_NUM","ELEMENT_NUM")
ENABLE,
CONSTRAINT "PRODUCES_SPLICED_TRANSCRI_FK1" FOREIGN
KEY ("ID_GEN", "ALLELE_NUM", "SPLICED_TRANSCRIPT_NUM")
REFERENCES "SPLICED_TRANSCRIPT" ("ID_GENE", "ALLE-
LE_NUM", "SPLICED_TRANSCRIPT_NUM") ENABLE) ORGA-
NIZATION INDEX NOCOMPRESS PCTFREE 10 INITRANS 2 MAX-
TRANS 255
LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS
1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL
DEFAULT)
TABLESPACE "USERS" PCTTHRESHOLD 50;

Capítulo 5

Análisis de Contenidos a efectos de carga de la Base de Datos Genómica

Resumen: En este capítulo se describe el estudio y el resultado del análisis realizado a las bases de datos públicas más utilizadas por los biólogos, como son las bases de datos Gene, Nucleotide y PubMed del Centro Nacional de Información Biotecnológica (NCBI) y la Base de Datos de Mutaciones Humana (HGMD) principalmente, de igual forma se expone el análisis genérico realizado con el objetivo de analizar la información que almacenan y su estructura. Una vez han sido analizadas las bases de datos se deduce un esquema conceptual para cada base de datos con el objetivo de comparar estos esquemas con el Esquema del Genoma Humano, a fin de determinar la información útil y relevante que estas bases de datos contienen con el objetivo principal de realizar la carga de la base de datos genómica descrita en el capítulo 6 de esta tesis.

5.1. Análisis sobre la estructura e información de la base de datos de referencia NCBI

El centro nacional de información biotecnológica (NCBI), el cual se establece en 1988 como un recurso nacional para la información de la biología molecular con el fin de mejorar la comprensión de los procesos moleculares que afectan a la salud humana y la enfermedad. El NCBI crea bases de datos públicas, lleva a cabo la investigación en biología computacional, desarrolla herramientas de software para el análisis de datos del genoma, y divulga información biomédica.

La principal función de NCBI es desarrollar nuevas tecnologías de información que ayuden en la comprensión de los procesos genéticos y moleculares que controlan la salud y la enfermedad. NCBI se encarga de crear sistemas automatizados para el almacenamiento y análisis de conocimientos de biología molecular, bioquímica y genética, además facilita el uso de estos repositorios y software para la investigación y la comunidad médica. De igual forma, recopila información biotecnológica a nivel mundial y realiza investigación en métodos computacionales avanzados de procesamiento de información para el análisis de la funcionalidad y la estructura biológica de las moléculas.

En cuanto a las bases de datos de secuencias de ADN, NCBI asume la responsabilidad de la base de datos GenBank en octubre de 1992 construyendo una base de datos a partir de secuencias que suministraban laboratorios particulares, intercambiando información con bases de datos internacionales como el Laboratorio de Biología Molecular Europeo (EMBL) y la Base de Datos de ADN de Japón (DDBJ).

La estructura de NCBI se divide en tres ramas organizacionales, la rama de la Computación Biológica (CCB), la rama de la Ingeniería de la Información (IEB) y la rama de las Fuentes de Información (IRB).

La rama de la Computación Biológica (CCB) es la que se encarga de llevar a cabo investigación básica y aplicada en informática, matemáticas y problemas

teóricos de la biología molecular y la genética, incluyendo el análisis del genoma, la comparación de secuencias, metodologías para la búsqueda de secuencias, estructuras macromoleculares, dinámica e interacción así como la predicción estructura/función.

La rama de la Ingeniería de la Información (IEB) realiza investigación aplicada en representación de datos y análisis, incluido el desarrollo de sistemas de cómputo para el almacenamiento, manejo y recuperación de los conocimientos relacionados a la biología molecular, genética y bioquímica. A su vez diseña esquemas de base de datos y especificaciones para la representación de las diversas formas de la información de la biología molecular e información estructural. De igual forma diseñan y desarrollan sistemas software, que proveen a los investigadores servicios computacionales remotos y locales.

La rama de las Fuentes de la Información (IRB) tiene la función de planificar, dirigir y gestionar las operaciones técnicas de NCBI, incluidos los sistemas informáticos utilizados para la investigación y desarrollo, así como los sistemas informáticos utilizados para acceder a bases de datos públicas.

NCBI tiene una estructura de gran escala a nivel de bases de datos, incluyendo las principales bases de datos utilizadas en el dominio biológico. Cuenta con bases de datos de literatura médica, genética y biológica, como *PubMed* y *OMIM*, también cuenta con base de datos moleculares como bases de datos de secuencias de nucleótidos (*dbEST, PopSet, dbGSS, Probe, dbSNP, RefSeq, dbSTS, SRA, Nucleotide, TPA, GenBank, Trace, Archive, HomoloGene, UniGene, MGC, UniSTS*) y de proteínas (*3D Domains, PROW Proteins, RefSeq, Protein Clusters*), bases de datos de estructuras proteómicas (*Conserved Domains, Structure (MMDB), 3D Domains*), Genomas Completos (*Cancer Chromosomes, Genome Project, COGs Genomes y Gene*) y Taxonomías (*Taxonomy*).

De todas las bases de datos que NCBI contiene, este trabajo se ha centrado en tres bases de datos: *Entrez Gene, Nucleotide* y *PubMed:*

Entrez Gene [7] es una base de datos de información génica orientada a genomas completamente secuenciados. El contenido de *Entrez Gene*, es el

resultado de la conservación y de la integración automatizada de los datos del proyecto de secuencia de referencia de NCBI (RefSeq) y de otras bases de datos disponibles en NCBI. La información de esta base de datos incluye la nomenclatura, localización en el cromosoma, productos del gen y sus atributos, marcadores asociados, fenotipos, interacciones, y enlaces a referencias, secuencias, detalles de variaciones, informes de expresión, homologías, proteínas y bases de datos externas. El objetivo principal de *Entrez Gene*, es proporcionar a los genes un identificador de tipo entero (GeneID) de seguimiento único y proporcionar información asociada a dichos identificadores para uso público.

NCBI ofrece diferentes formas de acceso a la información de sus bases de datos entre ellas *Entrez Gene*. La principal forma es enviar una consulta a través de la página Web de NCBI y mostrar los resultados en Gene. Como resultado de la consulta *Entrez Gene* provee múltiple informes, el primero de ellos es un vista resumen de la consulta, este resumen incluye las especies de origen, símbolos preferidos y alternativos (otros alias), nombres descriptivos preferidos o alternativos, localización en el cromosoma y el identificador del gen (GenID). El símbolo del gen es un enlace al informe completo del Gen, de igual forma, se incluyen enlaces generales a otros sitios (Ver Fig. 5.1)

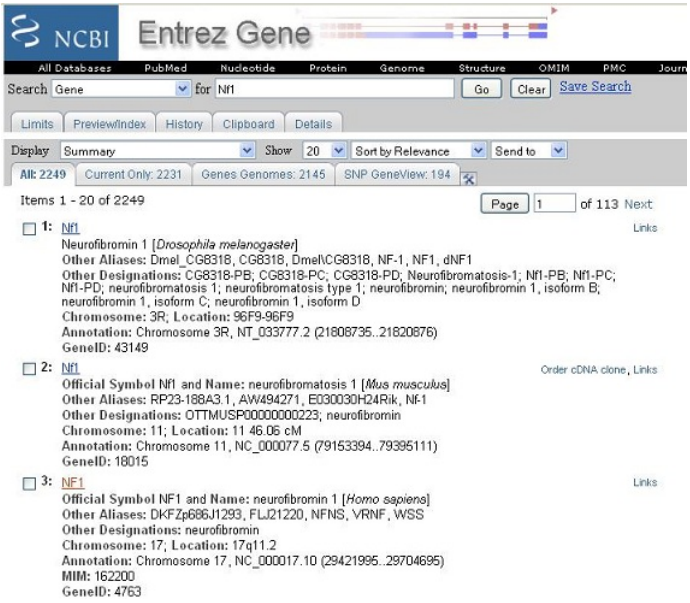


Figura 5.1: Resumen de resultados de una consulta en *Entrez Gene*

Toda la información que se provee de un Gen es definido en un archivo ASN.1 [8]. El informe completo de una consulta de un gen muestra esta información realizando la transformación de ASN.1 a HTML, incluyendo, herramientas de navegación, diagramas y textos. Alguna información no es mantenida por Entrez Gene pero sí por otras bases de datos de NCBI y bases de datos externas, por lo que el informe completo del gen provee un menú de enlaces en la parte superior derecha de éste.

El informe completo se divide en las siguientes secciones:

- Menús de navegación
- Título
- Resumen
- Regiones Genómicas, Transcripciones y Productos
- Contexto Genómico

- Bibliografía
- Interacciones
- Alelos
- Información General del Gen
- Información General de la Proteína
- Secuencias de Referencia de NCBI (RefSeqs)
- Secuencias Relacionadas
- Enlaces adicionales

Para el propósito de este trabajo, las subcategorías que interesan debido a su contenido son *Resumen, Regiones Genómicas, Transcripciones y Productos, Contexto Genómico, Bibliografía y Secuencias de Referencia de NCBI.*

En la subcategoría *Resumen*, se encuentra información descriptiva del gen, como el símbolo oficial y nombre, el identificador y enlace a los principales recursos fuera de NCBI que proporcionan información acerca del gen, esta información son dados por HGNC (Comité de Nomenclatura de Genes de la Organización del Genoma Humano), además se encunetra información sobre el locus del gen, el tipo de gen (*tRNA, rRNA, snRNA, scRNA, snoRNA, miscRNA, Codificador de Proteína, pseudo, otro, y desconocido*), el estado que se encuentra el gen respecto a RefSeq, el organismo al que pertenece el gen, el linaje del gen, los símbolos no oficiales y descripciones que se han utilizado para este gen y sus productos, y por ultimo un texto descriptivo sobre el gen, su localización celular, su función, y su efecto en el fenotipo (Ver Fig. 5.2).

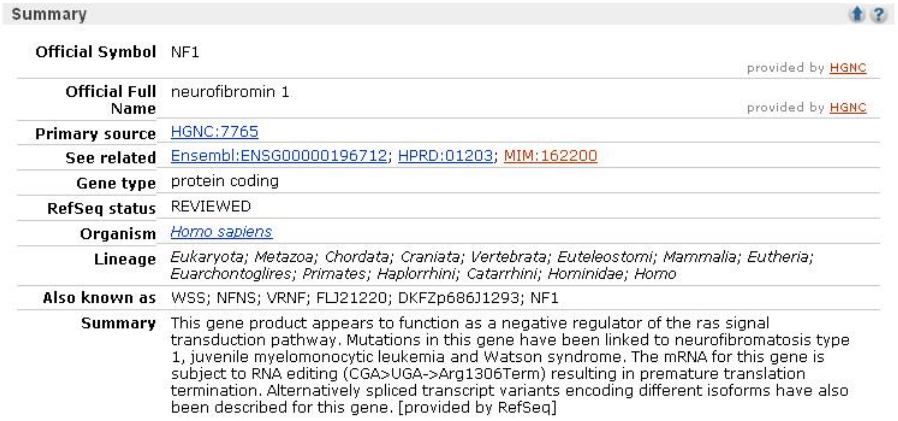


Figura 5.2: Sub - Categoría Resumen del Informe Completo *Entrez Gene*

En la subcategoría *Regiones Genómicas, Transcripciones y Productos*, se encuentra información sobre la posición de un pseudogene, del intrón, exón y la región de codificación siempre y cuando esta información esté disponible en los sistemas de coordenadas genómicos. Esta sección se utiliza para ver la organización de un gen (intrones, exones y región codificante), su producto ARN y la correspondiente posición en un genoma de referencia (RefSeq). También se utiliza para identificar las secuencias de referencia (RefSeq) correspondientes a su ARN o proteína producida y para tener una visión general de los exones que representan. A su vez, se utiliza para obtener las secuencias genómicas, de ARN y proteínica correspondiente, brindando un enlace a la base de datos *Nuclotide* de NCBI (Ver Fig. 5.3).

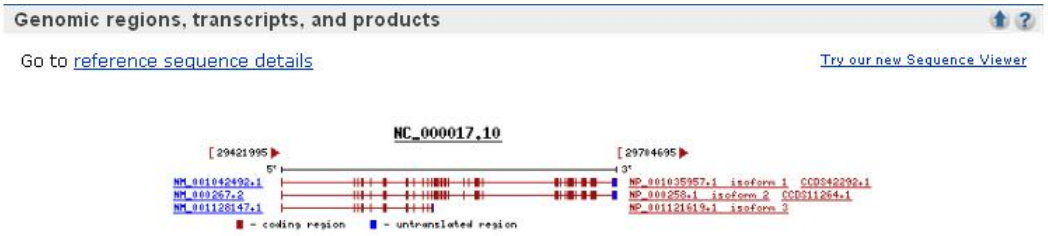


Figura 5.3: Subcategoría Regiones Genómicas, Transcripciones y Productos

En la subcategoría *Contexto Genómico*, se muestra la localización de un gen

en el cromosoma si el gen está en una anotación genómica. En esta sección se muestra un diagrama de los genes vecinos al gen y sus orientaciones.

La subcategoría Bibliografía, incluye enlaces a la base de datos de literatura (PubMed) de NCBI correspondientes al gen.

Y por último, en la subcategoría *Secuencias de Referencia* de NCBI (RefSeqs), se describe la secuencia de referencia específica de NCBI establecida para dicho gen y se proveen enlaces a las bases de datos relacionadas con el gen en NCBI. Además esta sección incluye descripciones de cada variante transcrito (ARNm) su correspondiente número de acceso y su correspondiente proteína codificada. En esta sección su utilizan diferentes aproximaciones para las secuencias de referencias y se dividen en dos grupos:

- *Secuencias de Referencia* mantenidas independientemente de los Genomas anotados: las secuencias de ARN y de las proteínas se actualizan continuamente independientemente de cualquier anotación de un genoma. Dado que estas secuencias son conservadas independientemente del ciclo de anotación del genoma sus versiones no coinciden con la versión actual de la secuencia de Referencia (RefSeq) del genoma secuenciado actual (Ver Fig. 5.4).

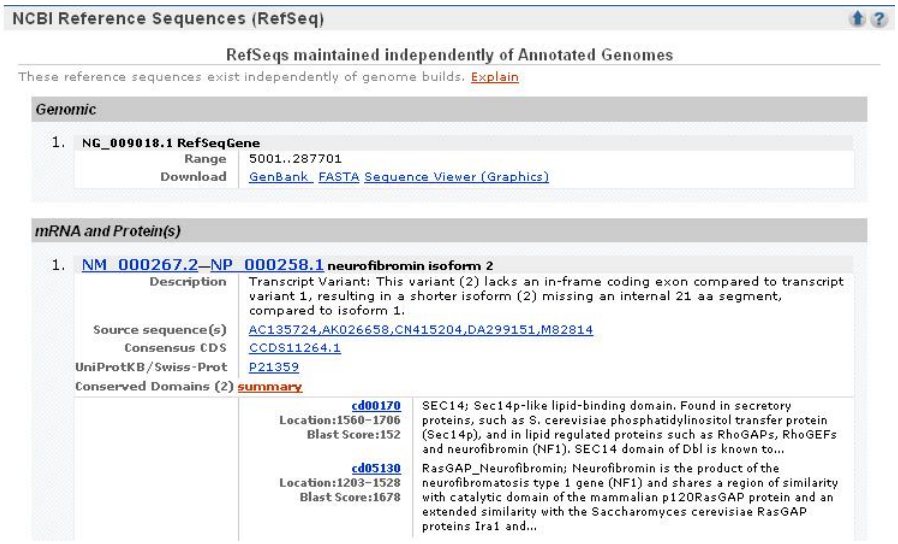


Figura 5.4: Secuencias de Referencia independientes de los Genomas Anotados

- *Secuencias de Referencia de Genomas Anotados*: se indican las Secuencias de Referencia (RefSeqs) de todos los ensamblajes en los que el gen es anotado, desde el de referencia hasta ensamblajes alternativos. Además se proveen enlaces a otras bases de datos para poder obtener sus secuencias y descripciones (Ver Fig. 5.5).

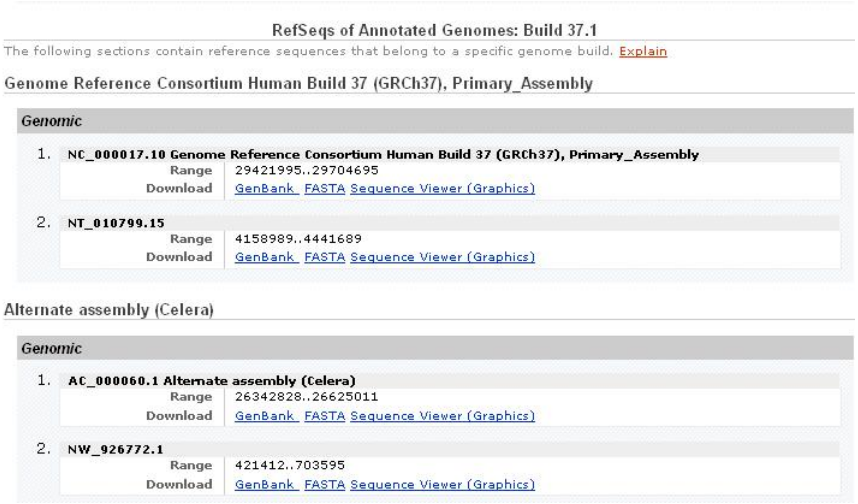


Figura 5.5: Secuencias de Referencia de Genomas Anotados

Otra de las bases de datos que es importante para este trabajo es *Nucleotide* de NCBI, la cual es una colección de secuencias de varias fuentes, incluyendo GenBank, RefSeq, y el PDB. De aquí se extraerán las secuencias de referencia de los genes y las secuencias de sus variantes (ARNm).

Para acceder a la información de la secuencia de referencia de un gen, en la subcategoría *Secuencias de Referencia* de NCBI del informe completo descrito anteriormente, existen enlaces correspondientes a la base de datos *Nucleotide*, así mismo con las variaciones (ARNm) y con las proteínas del gen.

Cuando se accede a la secuencia de referencia de un determinado gen, toda la información es mostrada en un informe, donde la primera parte corresponde a la descripción del gen (locus, definición y número de acceso), la segunda parte corresponde a las referencia bibliográficas del gen brindando enlaces a las base de datos de literatura Pubmed. La tercera y más importante parte, corresponde a la parte estructural del gen, donde se divide el gen en exones con su posición dentro del gen y en STS, y finalmente se provee toda la secuencia completa del gen.

Igualmente se pueden consultar las variaciones correspondientes al gen. La primera parte del informe corresponde a una descripción general de la variación, la segunda parte a referencias bibliográficas y enlaces a Pubmed, y la tercera parte corresponde a la descripción estructural de la variación, dividida por los correspondiente exones y proporcionando la correspondiente secuencia de la variación. Además, si se desea conocer o capturar la correspondiente secuencia de cada uno de los exones de la variación, existe un enlace que lleva a otro informe similar que los descritos anteriormente correspondiente al exón.

Una vez se ha analizado la base de Datos de referencia de NCBI, se procede a analizar la base de datos de referencia de mutaciones de HGMD.

5.2. Análisis sobre la estructura e información de las bases de datos de referencia HGMD

La Base de Datos de Mutaciones de Genes Humanos (HGMD)[9], es una gran colección de datos sobre mutaciones en la línea germinal de los genes nucleares asociados a enfermedades humanas, además representa una fuente de referencia actualizada del espectro de lesiones heredables en genes humanos. Fue desarrollada originalmente para el estudio de mecanismos mutacionales en genes humanos. HGMD posee todas las mutaciones de la línea germinal que producen enfermedades y los polimorfismos, tanto funcionales como los asociados a enfermedades, descritos en la literatura, y provee estos datos en un formato de fácil acceso para todos aquellos interesados, ya sean del ámbito académico, clínico o comercial. Es considerada la principal base de datos de mutaciones humanas asociadas a enfermedades disponibles para la comunidad científica [10]. En diciembre de 2008, la base de datos contenía más de 85.000 lesiones diferentes detectadas en 3.253 genes diferentes, actualmente se almacenan nuevas entradas a un ritmo superior a 9.000 por año. Los datos que se encuentra en HGMD comprenden desde sustituciones de una base en las diferentes regiones del gen (codificante, reguladora, splicing), micro–borrados, micro–inserciones, combinación de borrados e inserciones (indels), expansiones repetidas, grandes lesiones (borrados, inserciones y duplicaciones) y reordenamientos complejos. Los datos se almacenan en la base de datos semanalmente con procedimientos de búsqueda manuales y computarizados de registros. Más de 250 revistas se analizan en busca de artículos que describan mutaciones de línea germinal, que causan la enfermedad genética humana. Los datos requeridos se extraen de los artículos originales y se aumenta dicha información con los datos de apoyo necesarios. Los datos incluidos son principalmente de los informes publicados originalmente, aunque algunos datos se han tomado de actualizaciones de mutaciones y revisiones de artículos.

La información se accesible a través de Internet. Existen varias opciones para buscar las mutaciones de un gen, éstas son: por el símbolo del gen, por la

descripción del gen, por el número en OMIM, por el número de GDB, o por enfermedad/fenotipo. El resultado de la búsqueda arroja una interfaz donde se muestran los diferentes genes y un enlace (símbolo del gen) para ver las mutaciones correspondientes (Ver Fig. 5.6).

Search result for *'nf1'* using *gene symbol* search

Please click on the gene symbol to proceed to the relevant HGMD entry...

Gene symbol	Gene description	Location
NF1	Neurofibromatosis 1 protein (neurofibromin)	17q11.2

Figura 5.6: Resultado búsqueda en HGMD

Cuando se sigue con el enlace para acceder a la información relevante, es decir, las mutaciones, se muestra una lista de los tipos de mutaciones existentes para el gen, el número total de cada tipo de mutación y un enlace para poder ver dichas mutaciones. Además proveen un enlace, para obtener la secuencia de referencia de ADN copia (Ver Fig. 5.7).

Gene Symbol	Chromosomal location	Gene name	cDNA sequence	Extended cDNA	Splice junctions	Mutation
NF1	17q11.2	Neurofibromatosis 1 protein (neurofibromin)	Get cDNA	BIOBASE Feature available to subscribers	Splice junctions	BIOBA Feature available t

Mutation type	Number of mutations	Mutation data by type (register c
Missense/nonsense	220	Get mutations
Splicing	160	Get mutations
Regulatory	0	No mutations
Small deletions	237	Get mutations
Small insertions	113	Get mutations
Small indels	14	Get mutations
Gross deletions	77	Get mutations
Gross insertions	8	Get mutations
Complex rearrangements	8	Get mutations
Repeat variations	0	No mutations
Public total (HGMD Professional 2009.2 total)	**837** (1201)	

Figura 5.7: Listado de tipos de mutaciones de un gen

Una vez son listadas todos los tipos de mutaciones registradas en la base de

datos para dicho gen, existe enlaces correspondientes a cada tipo de mutación, donde finalmente se listan las mutaciones (Ver Fig. 5.8).

Accession Number	Description	Phenotype	Reference
CN921142	Insertion of 10 kb (described at genomic DNA level)	Neurofibromatosis 1	Upadhyaha (1992) Hum Mol Genet 1, 735
CN044486	Duplication of 23 bp c.5556-5578 (described at cDNA level)	Neurofibromatosis 1	De Luca (2004) Hum Mutat 23, 629
CN931362	Duplication of 42 bp cd. 1699-1713 (near perfect) (described at genomic DNA level)	Neurofibromatosis 1	Tassabehji (1993) Am J Hum Genet 53, 90
CN005225	Insertion of 74 bp from intr. 25, nt 4247 (described at genomic DNA level)	Neurofibromatosis 1	Fahsold (2000) Am J Hum Genet 66, 790

Figura 5.8: Ejemplo de lista de mutaciones de tipo Inserciones Gruesas

5.3. Matchings de la Bases de Datos Existentes

A continuación, a partir de los análisis de las bases de datos fuente, se deducirán los esquemas conceptuales tanto de NCBI como de HGMD, para así, posteriormente, realizar un matching de información entre dichos esquemas deducidos y el esquema conceptual del genoma humano.

5.3.1. Esquema Conceptual deducido de NCBI y matching con el ECGH

A partir del anterior análisis de cómo está estructurada la información en NCBI, se procede a deducir el Esquema Conceptual de la base de datos Gen de NCBI, para realizar una comparación con el ECGH con el objetivo de determinar qué información es relevante e importante tener en el ECHG, además, para tener conocimiento de que información existe y es adquirible actualmente para la posterior carga de la base de datos genómica. El esquema deducido se puede ver en la figura 5.9

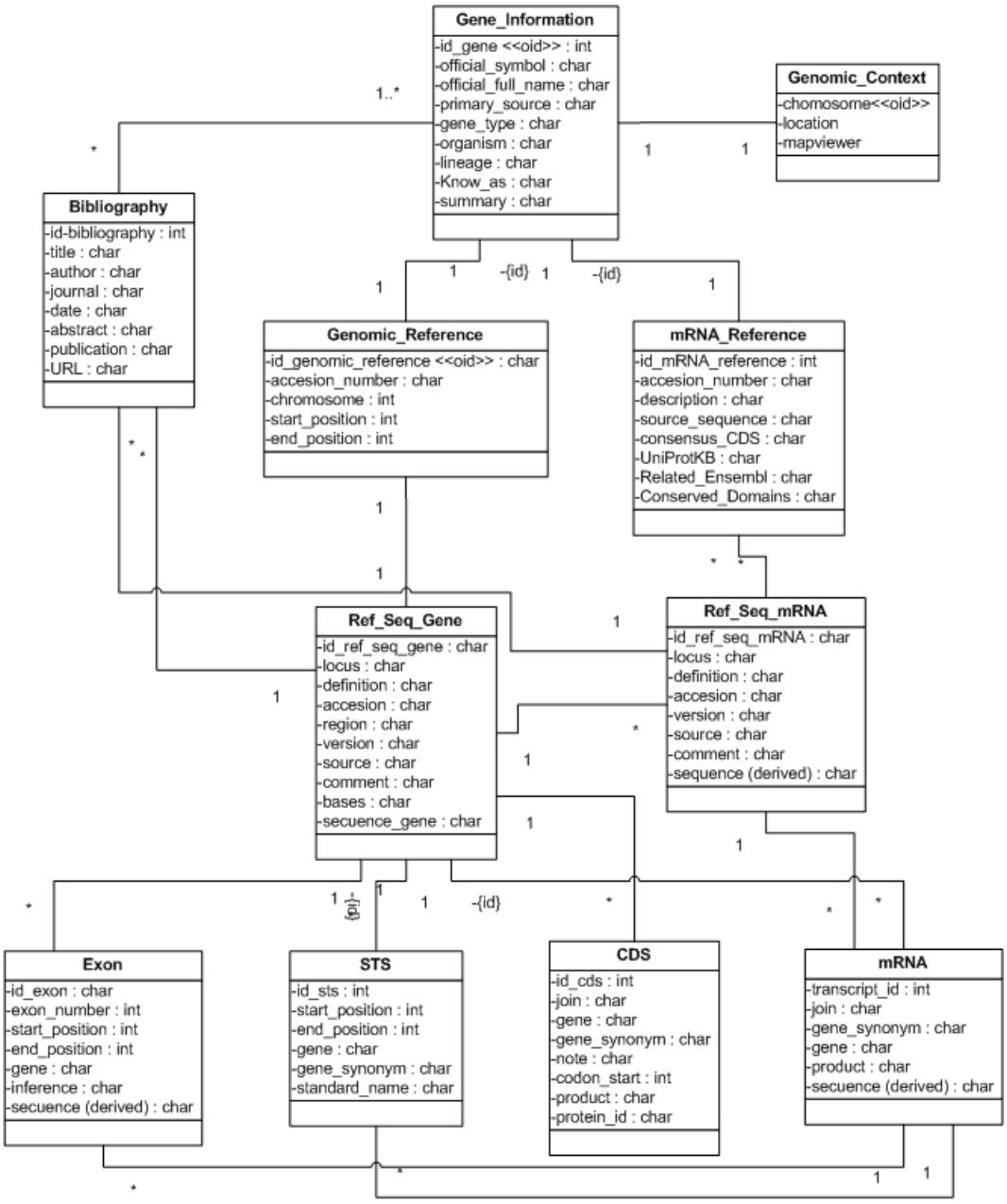


Figura 5.9: Esquema Conceptual deducido de NCBI

Es importante destacar que este esquema conceptual se centra más en la información básica del gen y de sus isoformas (mRNA's) omitiendo información irrelevante para los objetivos de este trabajo.

Por tanto, en la clase *Gene_ Information* del esquema conceptual deducido de NCBI (ECDNCBI) se tiene toda la información general de gen con sus atributos característicos.

Se tiene la clase *Genome_ context*, la cual especifica el cromosoma y su localización dentro del cromosoma del gen.

La clase *Genomic_ Reference* específica en qué posición dentro del cromosoma se encuentra el Gen, así como el número de acceso del Gen en la base de datos de NCBI.

La clase *mRNA_ Reference* específica el accesión number del mRNA, y he información relativa al ARN mensajero, como la descripción, la fuente de la secuencia, ensamblajes relativos, entre otros.

La clase *Ref_ Seq_ Gene*, tiene como atributos información sobre el gen, como el locus, la definición del gen, la versión del gen, el organismo o especie a que pertenece el gen, y el mas importante de todos, la secuencia de referencia del gen.

La clase *mRNA_ Seq_ Gene*, tiene atributos descriptivos, como el locus, la definición del ARN mensajero, la versión, el organismo el numero de y también la secuencia de el mRNA. La clase Exon, tiene como atributos, el numero del exón dentro del gen, la posición inicial y final del exón y el nombre del gen al que pertenece.

La clase *STS* representa Las diferentes STS (Sequence Tagged Site) del gen, con sus posiciones relativas dentro de la secuencia completa del gen, el nombre estándar y su enlace con la base de datos UniSTS.

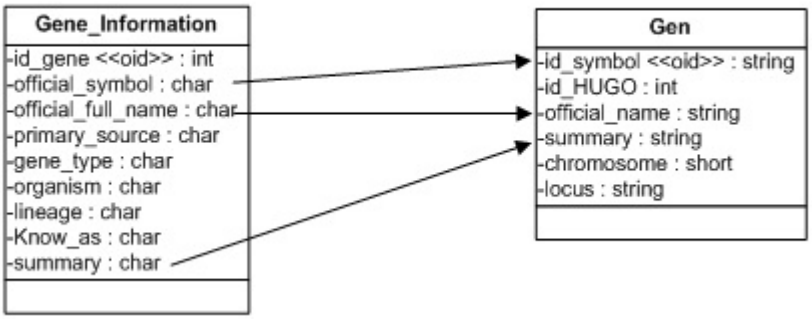
La clase *CDS* representa la secuencia consenso o CDS la cual establece las subsecuencias de la secuencia completa cuya unión da como resultado la secuencia transcribible del gen. Este tiene también como atributos el conjunto de posiciones inciales y finales dentro del gen del CDS, el codón inicial, notas relativas, el nombre del gen al que pertenece y el producto.

La clase *mRNA* tiene la información y los rangos o subsecuencias de la secuencia completa que componen el ARNm así como su secuencia.

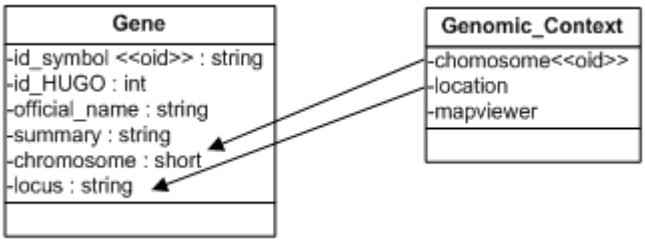
Finalmente la clase *bibliography*, representa todas las refernecias bilviograficas tanto del gen como del Arnm. Esta clse tien como atributos el titulo del a plublicacion, el o los autores, la revista donde se ha publicado, la fecha, el abstract y la publicación en sí.

Analizando el ECDNCBI, se deducen las siguientes correspondencias respecto al esquema conceptual del genoma Humano.

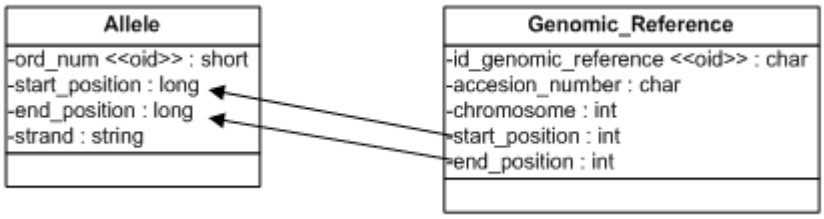
- La clase *Gene_ Information* se relaciona directamente con la clase *Gene* del ECGH, y así como los siguientes atributos: o*fficial_ symbol/id_ symbol, oficial_ fullname/oficial_ name, summary/summary.*



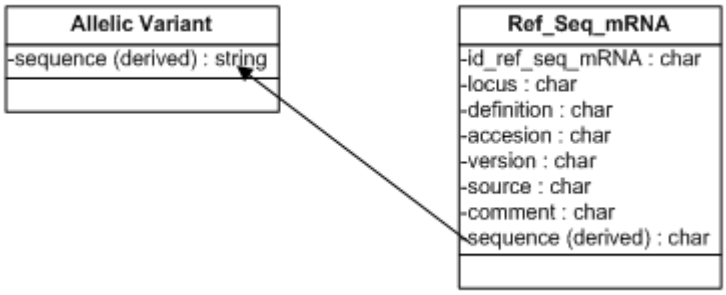
- De la clase *Genome_ context* los atributo chromosome y location, se relacional con los atributos de la clase *Gene* chromosome y locus directamente.



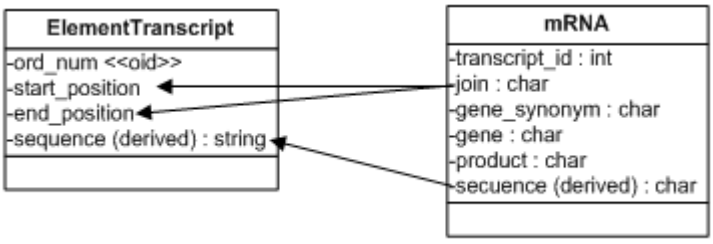
- De la clase *Genomic_ Reference* los atributos *start_ position y end_ position* se relacionan directamente.



- De la clase *Ref_ Ser_ mRNA*, el atributo *sequence* se relaciona con el atributo *sequence* de la clase *Allelic Variant.*



- De la clase *mRNA* se relacionan los atributos *join* con *start_ position y end_ position*, asi como el atributo *sequence* directamente.



- Y finalmente la clase *Exon* se relaciona directamenet con la clase *Exon* del ECGH. y la clase *Spliced_ Transcript* del ECGH se relaciona con la clase *Exon* y *mRNA* del esquema deducido de NCBI.

5.3.2. Esquema Conceptual deducido de HGMD

A partir del análisis de cómo está estructurada la información de HGMD visto en el punto 5.2 de esta tesis, se ha realizado la tarea de deducir un esquema conceptual de dicha información para poder compararlo con el esquema conceptual del Genoma Humano descrito en esta tesis, a fin de realizar mappings de información, y deducir la información más útil y relevante que se desea cargar en la base de datos genómica. El esquema conceptual resultante deducido para HGMD se puede ver en la figura 5.10.

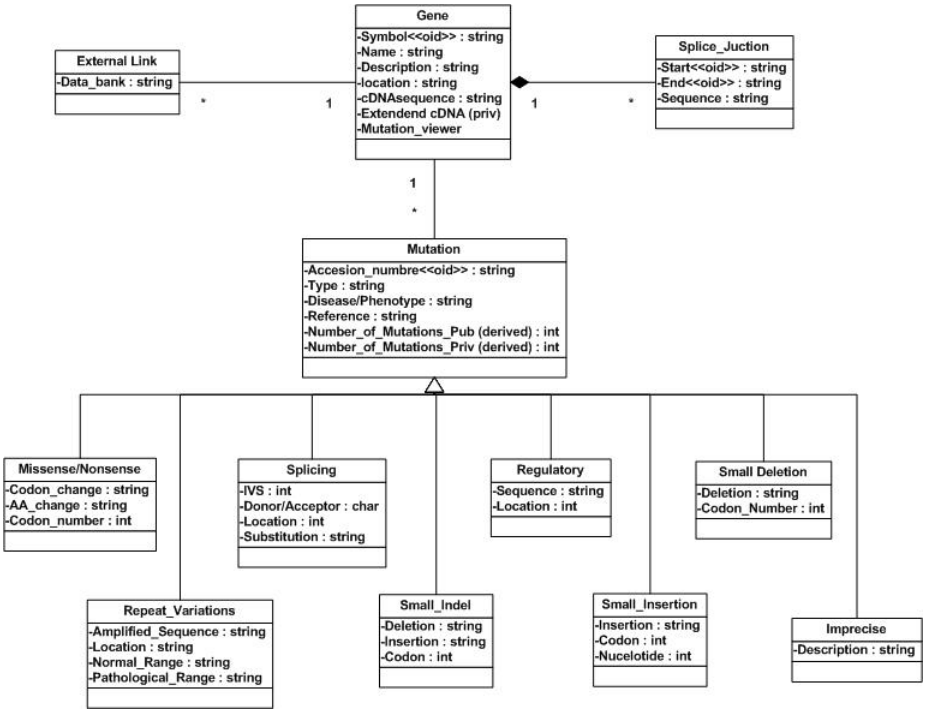


Figura 5.10: Esquema Conceptual deducido de HGMD

Cabe resaltar que la información que se desea encontrar en HGMD son las diferentes mutaciones de cada gen, por lo que la información propia del Gen no se tiene en cuenta de esta base de datos sino de GENE NCBI, por lo tanto, la comparación entre los dos esquemas conceptuales será solamente entre el esquema deducido de HGMD y la vista Mutaciones del Esquema Conceptual

del Genoma Humano. Sin embargo, conceptualmente se incluye esta clase en el modelo pues se inicia a partir de la información de un Gen, siendo esta clase una de las clases más importantes de todo el esquema conceptual deducido.

Esta clase tiene atributos de identificación como *Symbol*, el cual es el identificador del gen en la base de datos y el nombre del gen es incluido en el atributo *Name*, atributos de descripción como *Description, Location*, los cuales brindan una descripción general de la función del gen y la posición del gen en el cromosoma respectivamente. También se encuentra un enlace para obtener el ADNc correspondiente al gen, representado por el atributo *cDNAsequence*.

Otra clase importante en el esquema, es la clase *Mutation*. Esta clase tiene como atributo de identificación dentro de la base de datos *Accesion_ number*, atributos descriptores como *Type* para catalogar el tipo de mutación, *Disease/Phenotype* para describir la enfermedad o fenotipo asociado a la mutación, el atributo *Reference* muestra la referencia literaria donde la mutación ha sido descrita por primera vez. Otros dos atributos derivados *Number_ of_ mutations_ Pub y Number_ of_ mutations_ Priv* proporcionan información acerca del número total de mutaciones públicas y privadas existentes en la base de datos del respectivo gen.

Dependiendo del tipo de la mutación, se crean clases especializadas de la clase *Mutation*. La clase especializada *Missense/Nonsense* representa la mutación en la cual un nucleótido cambia, dando como resultado un codón diferente y por lo tanto codificando un aminoácido diferente (*Missense*) o en un codón de parada (*Nonsence*) admitiendo que se produzca una proteína no funcional. La mutación es descrita a partir de los atributos *Codon_ change*, el cual presenta el codón original y el codón resultante después de la mutación, el atributo *Aminoacid_ Change* muestra el par de aminoácidos codificados por el codón original y el codón mutado respectivamente y el atributo *Codon_ number* indica donde está localizado el codón afectado.

La clase *Splicing* describe mutaciones que están implicadas en el proceso de corte y empalme del ARN mensajero. Presenta la localización relativa donde

se presenta la lesión en el atributo *Location* con respecto al intrón enumerado (atributo *IVS*) en la parte donante o aceptora del splicing representado por el atributo *Donor/Acceptor*, y por último la base sustituida es representada por medio del atributo *Substitution*.

La clase *Regulatory* representa mutaciones causadas por sustituciones en las secuencias reguladoras, el atributo *Sequence* representa las sustituciones registradas mientras que el atributo *Location* representa la localización de la mutación respecto al sitio de iniciación de la transcripción, al codón de iniciación o al codón de terminación.

Los borrados de 20 pares de bases o menos en una secuencia son considerados como pequeños borrados. Esto se representa con la clase *Small Deletion*. El atributo *Deletion* contiene la secuencia con las bases borradas en minúsculas seguida del resto de la secuencia en Mayúscula, cuenta con 10 pares de bases que rodean entre en ambos lados de la lesión en la secuencia de ADN. El atributo *Codon_number* representa el último codón completo antes del borrado.

La clase *Small Insertion* representa inserciones de 20 pares de bases o menos en una secuencia de ADN. El atributo *Insertion* contiene la secuencia con las bases insertadas en minúsculas seguido de la secuencia en mayúsculas con un rango de 10 pares de bases a cada lado de la lesión. El número del codón donde ocurre la inserción es representado por el atributo *Codon_number* y el atributo *Nucleotide* describe la posición del primer nucleótido insertado.

La combinación de inserciones con borrados se clasifican como mutaciones *Indel*, y es representada con la clase *Small_Indel* en el modelo deducido, el atributo *Deletion* contiene la secuencia con las bases borradas en minúsculas seguido de la secuencia en mayúsculas con un rango de 10 pares de bases a cada lado de la lesión. Así mismo el atributo *Insertion*, contiene la secuencia con las bases insertadas en minúsculas seguida de la secuencia en mayúsculas.

Existen mutaciones (grandes borrados, grandes inserciones y reordenamientos complejos) que son catalogadas como imprecisas pues la información que

se encuentra de dichas mutaciones son considerablemente variables por lo que se realiza una descripción global de la mutación. Estas mutaciones son representadas en una sola clase (Clase *Impecise*) en el modelo deducido pues comparten el mismo tipo de información. El atributo *Description* contiene información sobre la naturaleza y localización de la lesión de forma narrativa o dando una descripción general de la mutación.

La clase *Repeat_ Variations*, también describe una mutación imprecisa (variaciones repetidas) sin embargo, esta clase tiene otros atributos descriptivos que brindan un poco más de información sobre la mutación. El atributo *Amplified_ Sequence* contiene la secuencia repetida, los atributos *Location, Normal_ Range y Pathological_ ranges* presentan información de forma imprecisa y de forma narrativa sobre la localización, el rango normal y los rangos patológicos de la mutación respectivamente.

La clase *Splice_ junctions* representa la información de cada uno de los cruces de empalme de un gen. Esta información es presentada en HGMD como un texto único que contiene todos los cruces de empalme del gen. Este texto tiene una estructura definida, por lo que se puede deducir el inicio y el fin de cada cruce, este inicio y fin son representados por los atributos *Start* y *End*, así como la correspondiente secuencia del cruce, la cual consisten en aproximadamente 25 pares de bases de la secuencia del exón representada en mayúscula seguido de 25 pares de base de secuencia del intrón representada en minúscula. Esta secuencia es representada por el atributo *Sequence*.

Por último, la clase *External_ Link*, representa todos los enlaces asociados a cada gen que proveen información sobre dicho en gen en diferentes bases de datos externas a HGMD.

Capítulo 6

Análisis del prototipo de carga de la Base de Datos Genómica

Resumen: en este capítulo se describe la carga inicial de información de la base de datos genómica. No obstante, antes se analiza que métodos y mecanismos de carga se deben utilizar así como las herramientas que brindan las bases de datos de NCBI y HGMD. Posteriormente se analiza y se diseña el prototipo de carga y actualización de la base de datos genónica a partir de la base de Datos de NCBI.

6.1. Mecanismos de Carga de la base de datos genómica

Para desarrollar los mecanismos de carga de la base de datos genómica, se debe primero buscar, analizar y probar que herramientas existentes pueden ser útiles para esta tarea. Después de este análisis se debe decidir y crear los mecanismos para la carga inicial de la base de datos. Para esto se analiza las herramientas que brinda NCBI y HGMD, pues son las bases de datos de referencia de este trabajo.

6.1.1. Herramientas de recuperación de datos de NCBI

Antes de analizar las herramientas que NCBI brinda, se debe saber cómo están clasificadas sus bases de datos. De acuerdo con el NCBI Site Map [11], las bases de datos que pueden explorarse desde el motor de búsqueda de Entrez pueden clasificarse en tres partes: Literature Databases, Molecular Databases y Genomes. De estas clasificaciones Molecular Databases se subdivide en Nucleotide Sequences, Protein Sequences, Structures, Genes, Gene Expresion y Taxonomy . El siguiente esquema (Ver Fig. 6.1) muestra a las bases de datos que se muestran en el Entrez Global Query de acuerdo con esta clasificación.

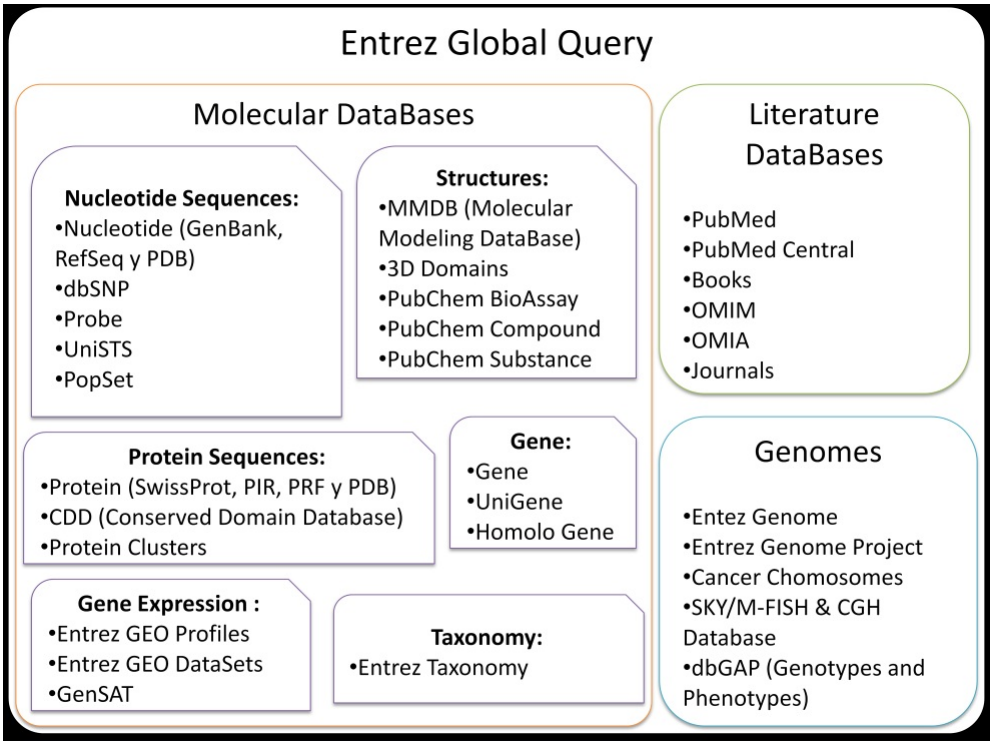


Figura 6.1: Clasificación de las base de datos en Entrez Global Query

El sistema Entrez que proporciona NCBI es un sistema integrado de búsqueda y recuperación de las bases de datos más importantes de NCBI incluyendo las siguientes bases de datos: PubMed, Nucleotide y Protein Sequences, Protein

Structures, Complete Genomes, Taxonomy, y otras más. El sistema Entrez esta dividió en dos partes:

- Herramientas a través de la Web:
  - *Batch Entrez:* sube un archivo que contiene los números de acceso que NCBI ha asignado a cada secuencia procesada para recuperar su información.
  - *PubMed Batch Citation Matcher:* envía información de referencias a Entrez y recupera los IDs de PubMed para vincular referencias desplegadas u otras aplicaciones.
  - *Advanced Entrez Searching:* técnicas de búsqueda avanzada para la Web Entrez
  - *My NCBI:* incluye correo electrónico automático de búsqueda de actualizaciones y los filtros para los resultados de búsqueda.
  - *ftp:* a través del ftp de NCBI, se puede bajar en formatos .gz[1] la información de cada una de las bases de datos de NCBI.
- Herramientas para desarrollar:
  - *E-Utilities:* son herramientas que facilitan el acceso a los datos de Entrez fuera de la interfaz web de consulta de NCBI y pueden ser útiles para recuperar resultados de búsqueda en otros entornos, como aplicaciones externas o propias de un usuario.
  - *E-Utilities Web Service* de NCBI permite a los desarrolladores acceder a dichas herramientas de recuperación y búsqueda a través de SOAP (Simple Object Access Protocol a través de tecnologías como C# y Visual Basic in MS Visual Studio 2008 o Java (Apache Axis2 version 1.5.2). Los servicios Web pueden trabajar con versiones anteriores de Axis (Axis para Java ver. 1.4) y MS Visual Studio (MSVS 2003). Los desarrolladores pueden utilizar

[1] http://www.gzip.org/

otras herramientas y bibliotecas SOAP para acceder a los servicios Web de Entrez Utilities de NCBI. Estos servicios web dan acceso a EGQuery, ESummary, EInfo, ELink, ESearch, ESpell, EPost **y** EFetch.

  - **EGquery**: proporciona información en XML de una búsqueda simple.
  - **ESummary**: proporciona un resumen de la documentación de una lista de ID's de revistas o registros de NCBI.
  - **EInfo**: proporciona los nombres de campo, la última actualización, y enlaces disponibles para cada base de datos de Entrez.
  - **ELink**: Comprueba la existencia de un vínculo o artículos relacionados externos, recupera y anota la relevancia de los enlaces a bases de datos de Entrez o artículos relacionados.
  - **ESearch**: Busca y recupera identificaciones primarios de los registros en NCBI (para su uso en EFetch, eLink y ESummary).
  - **ESpell**: Recupera sugerencias de ortografía, si está disponible.
  - **EPost**: Envía un archivo que contiene una lista de identificadores para futuras búsquedas.
  - **EFecth**: Recupera los registros de una lista de identificadores en un formato requerido por el usuario.

- *NCBI C++ Toolkit:* El NCBI C + + Toolkit proporciona un conjunto de librerías descargables y aplicaciones para ayudar a la ciencia genética. Este framework incluye librerías de red, librerías SQL y el acceso BerkeleyDB, CGI y manejo de HTML, ASN.1 y XML, sistemas de alineación de secuencias, sistemas de recuperación de secuencias, entre otras funcionalidades.

Después de analizar cada herramienta que brinda NCBI, se ha decidido implementar el modulo de carga inicial de la base de datos a través de los

servicios web (**E-Utilities Web Service**), por su facilidad de implementación e interoperabilidad entre lenguajes de programación o plataformas, además por la facilidad de comunicación para el intercambio de datos entre las aplicaciones, Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

6.1.2. Herramientas de recuperación de datos de HGMD

HGMD ofrece una suscripción HGMD ® Professional, el cual es una compilación de datos estructurados y revisados de forma manual con la literatura permitiendo un rápido acceso a las consultas sobre las mutaciones actualizado constantentemente. Cuenta con 108.046 mutaciones y polimorfismos asociados a enfermedades en 3.959 genes tomados de 32.246 artículos de revistas y proporciona 3.880 secuencias de DNA de referencia.

HGMD ® Professional es ampliamente utilizado en la investigación genética humana y en el desarrollo de aplicaciones genéticas. Estas son algunas de las muchas ventajas que ofrece HGMD ® Professional:

- Actualización constante de los datos de las mutaciones
- Cobertura total a la literatura de Pubmed.
- Vista de las mutaciones por enfermedad y fenotipo.
- Herramientas avanzadas de búsqueda.
- Links a la base de datos de Entrez de SNP's
- Los datos son descargables.

6.2. Metodología y Desarrollo de las Herramientas de Carga

A continuación se explica la metodología propuesta para el desarrollo del sistema de carga y actualización de la BDGH. Posteriormente, se analiza el desarrollo del prototipo del sistema de carga de la base de datos del Genoma Humano.

6.2.1. Metodología de Desarrollo

En primer lugar se debe establecer una metodología con el fin de desarrollar módulos para carga y actualización de la base de dato genoma a partir de diversas fuentes de datos. Para esto se requiere generar prototipos de cada base de datos e identificar los patrones, estableciendo una arquitectura que evolucione conforme se integran nuevas bases de datos con métodos de acceso diferentes o estructuras diferentes.

La metodología propuesta consiste en una serie de análisis con el fin de obtener los requerimientos para el diseño de los módulos de carga y actualización de la base de datos genoma. Para cada fuente se debe de hacer lo siguiente:

1. Análisis de las herramientas disponibles para el acceso a la base de datos
2. Análisis del proceso de extracción de los datos.
3. Análisis de estrategias de transformación de modelos de datos.
4. Análisis de las estrategias de carga.
5. Identificación de requerimientos
6. Análisis y diseño de los módulos.

7. Identificación de la estructura de datos de la fuente y su correspondencia con la estructura de la base de datos genoma.

8. Extensión al prototipo correspondiente o generación de un nuevo prototipo.

Para ver esta metodología de una forma más grafica y entendible se puede ver en la figura 6.2

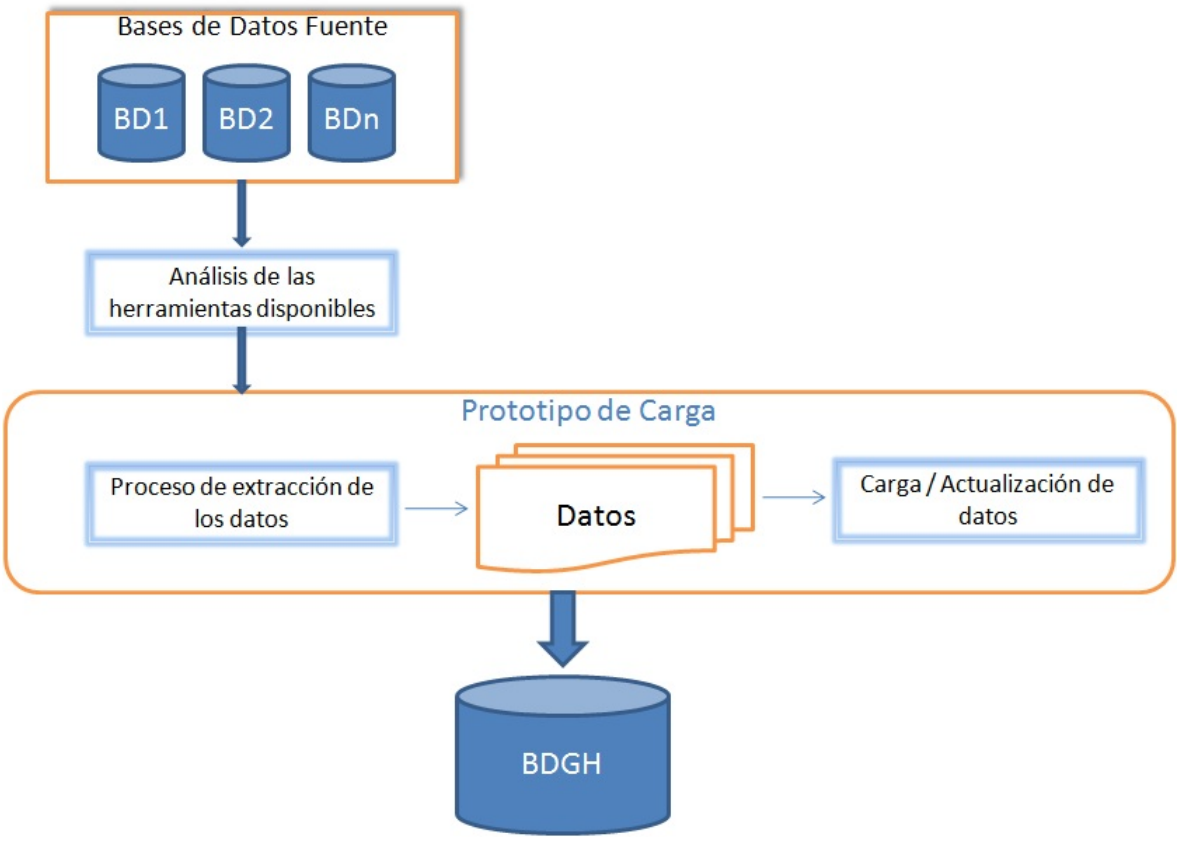


Figura 6.2: Proceso para la carga y actualización de la BDGH

Siguiendo los pasos anteriores para el desarrollo del prototipo de carga de la base de datos se da los siguientes resultados, teniendo en cuenta que se toma como referencia las bases de Datos de NCBI:

- Análisis de contenidos en las base de datos de la fuente: se analiza los atributos a cargar correspondientes a un registro dentro de cada tabla en la base de datos. Los atributos identificados son: LOG, GEN, BIBLIOGRAPHY_REFERENCE, BIBLIOGRAPHY_DATABANK, REFERENCE_GENE, ALLELE, REFERENCE_ALLELE, TRANSCRIPTION_UNIT, SEGMENT, REFERENCE_SEGMENT, SPLICED_TRANSCRIPT, REFERENCE_SPLICED_TRANSCRIPT.

- Análisis de las herramientas disponibles para el acceso a las bases de datos de la fuente: se utilizan los servicios Web de la interfaz Entrez Utilities para el acceso a los datos. Se seleccionaron los siguientes servicios Web de acuerdo con el Análisis anterior donde se identificaron los datos a extraer.

  - EUtils
  - eFetchGene
  - eFetchSeq
  - eFetchPubmed

- Análisis del proceso de extracción de los datos: se inicia con la extracción de los atributos identificados de la base de datos Gene mediante eFetchGene a un arreglo. Este arreglo contiene la información principal del gen y los enlaces a las bases de datos de secuencias y referencias correspondientes al gen.

- Identificación de la estructura de datos y su correspondencia con la estructura de la base de datos genoma:

  Se identificaron los siguientes vectores y se plantea su correspondiente método de transformación.

  - *Vectores intermedios*: son los vectores que corresponden una parte de un vector completo, o también pueden incluir varios vectores. Son utilizados durante la transformación. Se identifican por su descripción en la fuente: vector principal del gen, secuencia principal, ARN, exón, intrón y referencia.

- *Vectores auxiliares*: estos son vectores que corresponden a los identificadores de las referencias entre diversas bases de datos.

Los métodos de los vectores intermedios conforman la estrategia de transformación y los vectores auxiliares ayudan en el proceso de extracción de los vectores a transformar.

6.2.2. Análisis de los módulos a desarrollar del prototipo del sistema de carga de la BDGH

Se identifican cada uno de los módulos necesarios para el desarrollo del prototipo de carga de la base de datos y posteriormente se analizan para su desarrollo. Estos son los módulos identificados:

El primer modulo o modulo principal se denomina *Carga de lista de Genes*, el cual tiene tres submodulos: *Administrar lista de Genes, Obtener datos de cada gen de la Lista* y *Cargar Datos de cada gen de la lista.*

El submodulo *Administrar Lista de Genes* incluye otro submodulo que se denomina *Obtener el identificador del gen en NCBI.*

El submodulo *Obtener datos de cada gen de la Lista* incluye los submodulos: *Extraer los vectores del NCBI* y *Transformar los vectores de NCBI a vectores de Trasncription View.*

Por último el submodulo *Cargar Datos de cada gen de la lista*, tiene un submodulo denominado *Cargar datos de genes en Transcription View.*

A continuación se muestra los diagramas de Casos de Uso de UML de cada modulo describiendo las actividades principales a realizar por el sistema de cargar. En estos casos de uso también se muestran las dependencias que existen entre estos casos de uso y sus actores.

El usuario depende del *CU 1 Cargar lista de genes*, el cual incluye el *CU 1.1 Administrar lista de genes*, el *CU 1.2 Obtener los datos de cada Gen de la*

lista de la fuente NCBI y el *CU 1.3 cargar datos de cada gen de la lista* (Ver Fig. 6.3)

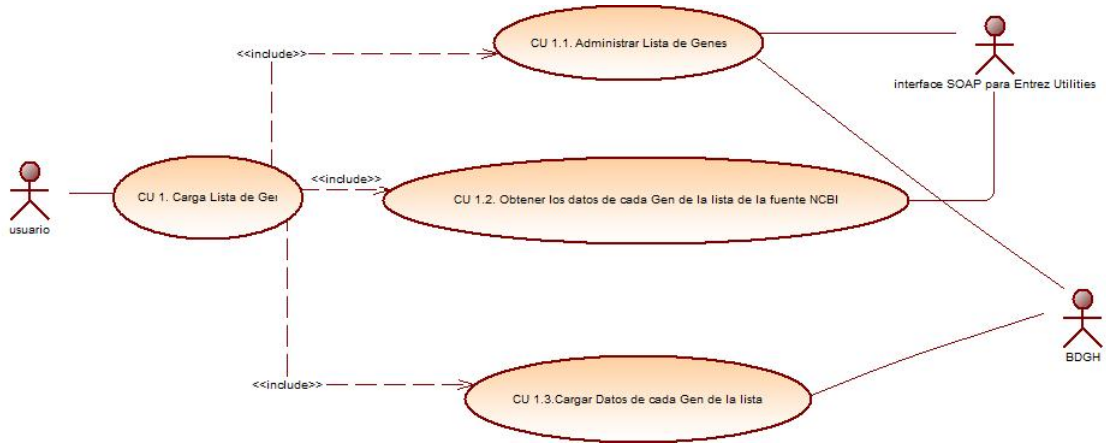


Figura 6.3: Caso de uso 1 (CU 1) nivel 0

En la figura 6.4 se muestran los casos de uso *Administrar lista de genes* para la carga de datos en la BDGH[2]. También se muestran las dependencias que existen entre estos casos de uso y sus actores. El módulo *Administrar lista de Genes* incluye el *CU 1.1.1 Obtener identificador del gen en el NCBI para Transcription View* el cual se relaciona con el servicio Web EUtils de la interfaz SOAP para Entrez Utilities y con la BDGH.

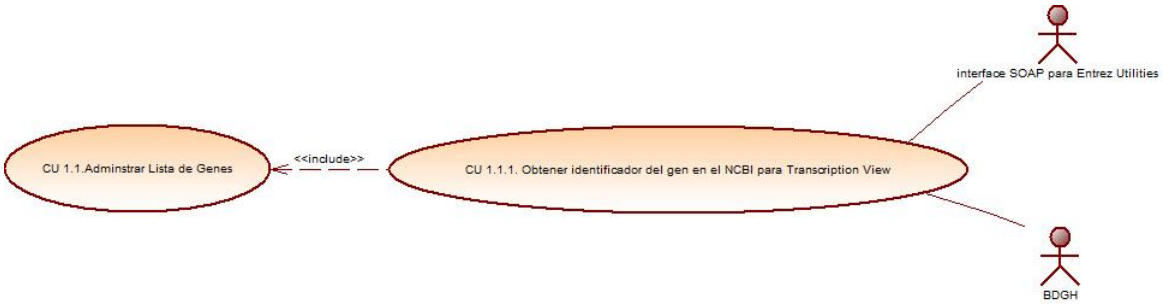


Figura 6.4: Caso de uso1.1 (CU 1.1) nivel 1

En figura 6.5 se muestran los casos de uso del módulo *Obtener datos de cada gen de la lista de la fuente NCBI*. También se muestran las dependencias que

[2]Base de Datos del Genoma Humano

existen entre estos casos de uso y sus actores. El módulo *Obtener datos de cada gen de la lista de la fuente NCBI* es extendido mediante el *CU 1.2.1 Extraer los vectores del NCBI* y el *CU 1.2.2 —Transformar los vectores de NCBI a vectores de Trasncription View*. El CU 1.2.1 se relaciona con los servicios Web: *eFetchGene, eFetchSeq* y *eFetchPubmed* de la interfaz SOAP para *Entrez Utilities*.

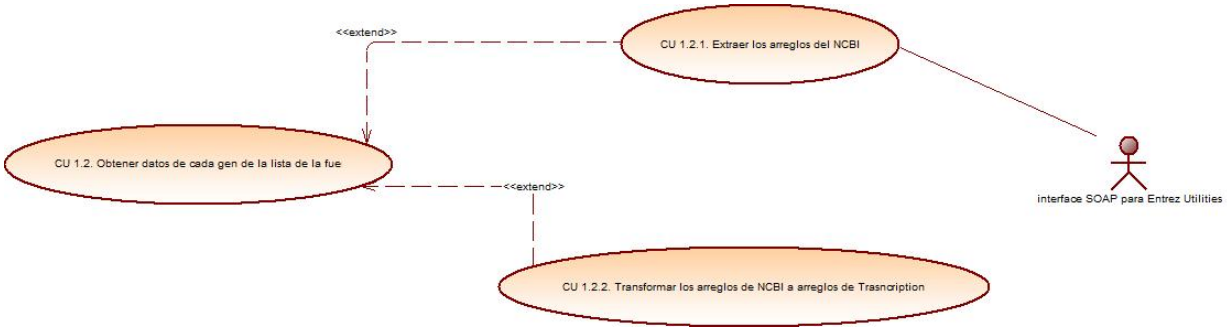


Figura 6.5: Caso de uso 1.2 (CU 1.2) nivel 1

Por último, En figura 6.6 se muestran los casos de uso del módulo *Cargar datos de cada gen de la lista* para BDGH. También se muestran las dependencias que existen entre estos casos de uso y sus actores. El módulo *Cargar datos de cada gen de la lista* es extendido mediante el *CU 1.3.1 Cargar datos de genes en Transcription View* el cual se relaciona con la DBGH.

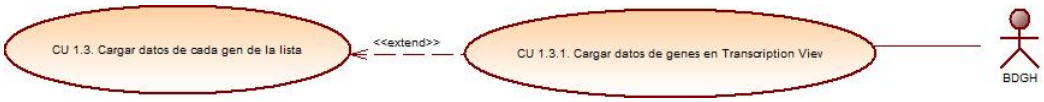


Figura 6.6: Cargar datos de genes en Transcription View de BDGH

6.2.3. Especificación de los casos de uso Prototipo de carga de BDGH

A continuación se especifican los casos de uso nombrados anteriormente.

6.2.3.1. Escenarios para el caso de uso "Cargar Lista de Genes"

ID	CU 1
Nombre Caso de Uso	Cargar Lista de Genes
Descripción:	Se encarga de obtener los datos de una lista de genes y almacenarlos en la base de datos *genoma*
Actores:	Usuario
Precondición:	Conexión estable a *Internet*. Conexión estable con la base de datos *genoma*. Servicios *Web* de *Entrez Utilities* están disponibles
Escenario Principal:	1. El módulo *cargar lista de genes* recibe la solicitud del usuario de generar una lista de genes mediante el apoyo del caso de uso 1.1. 2. El módulo *cargar lista de genes* recibe la solicitud del usuario de iniciar la obtención de los datos de la fuente *NCBI* mediante el apoyo del caso de uso 1.2. 3. El módulo *cargar lista de genes* informa al usuario acerca del estado actual del proceso en ejecución: obteniendo datos del gen y tipo de dato (vectores). 4. El módulo *cargar lista de genes* inicia el proceso de carga de los datos en la base de datos *genoma* mediante el apoyo del caso de uso 1.3. 5. El módulo de *cargar lista de genes* informa al usuario acerca del estado actual del proceso en ejecución: guardando datos del gen y tipo de dato (vectores). 6. El módulo de *cargar lista de genes* informa al usuario acerca del estado actual del proceso en ejecución: carga finalizada.
Prioridad	alta

6.2.3.2. Escenarios para el caso de uso "Administrar lista de genes"

ID	CU 1.1
Nombre Caso de Uso	Administrar Lista de Genes
Descripción:	Se encarga de crear y llenar una lista de genes con identificadores válidos.
Actores:	Usuario
Precondición:	Conexión estable a *Internet*. Conexión estable con la base de datos *genoma*.
Escenario Principal:	1. El módulo *administrar lista de genes* crea una lista nueva de genes. 2. El módulo *administrar lista de genes* llena la lista de genes con los datos obtenidos mediante el apoyo del caso de uso 1.1.1. 3. El módulo *administrar lista de genes* permite al usuario guardar la lista de genes. 4. El módulo *administrar lista de genes* guarda la lista de genes en un dispositivo de almacenamiento secundario. 5. El módulo *administrar lista de genes* informa al usuario de que la lista de genes ha sido guardada correctamente.
Prioridad	alta

6.2.3.3. Escenarios para el caso de uso "Obtener identificador del gen en el NCBI para Transcription View"

ID	CU 1.1.1
Nombre Caso de Uso	Obtener identificador del gen en el *NCBI* para *Transcription View*
Descripción:	Se encarga de obtener y validar cada elemento de la lista de genes.
Actores:	CU 1.1 Servicio *Web EUtils* de la interface *SOAP* para *Entrez Utilities* Base de datos *genoma*
Precondición:	Conexión estable a *Internet*. Interface *SOAP* para *Entrez Utilities* disponible. Acceso a la base de datos *genoma* disponible.
Escenario Principal:	1. El módulo obtener identificador del gen en el *NCBI* para *Transcription View* extrae el identificador del gen mediante el servicio *Web Eutils* de la interface *SOAP* para *Entrez Utilities*. 2. Si extrae el identificador del gen, el módulo *obtener identificador del gen en el NCBI para Transcription View* verifica los datos del gen obtenido en la tabla *LOG* de la base de datos *genoma*. 3. El *módulo obtener identificador del gen en el NCBI para Transcription View* devuelve un elemento validado al caso de uso 1.1.
Prioridad	alta

6.2.3.4. Escenarios del caso de uso "Obtener datos de cada gen de la lista de la fuente NCBI"

ID	CU 1.2
Nombre Caso de Uso	Obtener datos de cada gen de la lista de la fuente *NCBI*.
Descripción:	Se encarga de obtener los datos de una lista de genes y almacenarlos en la base de datos *genoma*.
Actores:	CU 1
Precondición:	Conexión estable a *Internet*. Lista de genes con identificadores Interface *SOAP* para *Entrez Utilities* disponible.
Escenario Principal:	1. El módulo *obtener datos de cada gen de la lista de la fuente NCBI* lee los datos de referencia de cada gen de la lista. 2. El módulo *obtener datos de cada gen de la lista de la fuente NCBI* extrae la información correspondiente de cada gen de la interface *SOAP* para *Entrez Utilities* mediante el caso de uso 1.2.1. 3. El módulo *obtener datos de cada gen de la lista de la fuente NCBI* transforma los datos de tipo *Transcription View* en los vectores *principal gene, secuencia principal, ARN, exón_intron* y *referencia* mediante el caso de uso 1.2.2. 4. El módulo *obtener datos de cada gen de la lista de la fuente NCBI* informa sobre la información obtenida al caso de uso 1.
Prioridad	alta

6.2.3.5. Escenarios para el caso de uso "Cargar datos de cada gen de la lista"

ID	CU 1.3
Nombre Caso de Uso	Cargar datos de cada gen de la lista.
Descripción:	Se encarga de almacenar la información referente a cada gen de la lista en la base de datos *genoma*.
Actores:	CU 1
Precondición:	Conexión estable a *Internet*. Base de datos *genoma*. Datos extraídos y transformados de cada gen.
Escenario Principal:	1. El módulo *cargar datos de cada gen de la lista* recibe cada vector de cada gen. 2. El módulo *cargar datos de cada gen de la lista* carga cada vector de cada gen en la base de datos *genoma* correspondiente a *Transcription View* (LOG, GEN, BIBLIOGRAPHY_REFERENCE, REFERENCE_GENE, ALLELE, REFERENCE_ALLELE, TRANSCRIPTION_UNIT, SEGMENT, REFERENCE_SEGMENT, SPLICED_TRANSCRIPT, REFERENCE_SPLICED_TRANSCRIPT) mediante el caso de uso 1.3.1. 3. El módulo *cargar datos de cada gen de la lista* devuelve la confirmación de que la inserción se realizó correctamente.
Prioridad	alta

Capítulo 7

Conclusiones

La primera conclusión que se puede extraer del trabajo realizado es el alto grado de dificultad que presenta el Genoma que es el conocimiento que se ha modelado. Llegar a comprender este dominio ha supuesto un reto importante para el grupo de trabajo. La obtención del ECGH ha sido y sigue siendo una tarea compleja y en evolución permanente.

El trabajo presentado en esta tesina parte de un estado del ECGH[5]evidenciando que el uso del Modelado Conceptual en cualquier dominio y más aun en el dominio de la Bioinformática, es un técnica de desarrollo fiable y muy ventajosa pues trabajar en un nivel alto de abstracción ya que permite que los conceptos y procesos sean mejor comprendidos.

A partir de este ECGH se ha realizado la transformación a un esquema de base de datos relacional. Para resultar más eficiente desde el punto de vista del desarrollo de aplicaciones, el esquema ha sido refinado.

La base de datos obtenida ha sido una herramienta eficiente para el diseño e implementación de un sistema de información que contribuye de manera eficiente a resolver los problemas de integración de datos para la búsqueda y recuperación de información valiosa acerca de los estudios realizados sobre la secuenciación de genomas de los seres humanos.

El análisis realizado a las fuentes de datos más utilizadas por los biólogos ha puesto de manifiesto la gran diversidad existente y la ausencia total de estándares en cuanto a la definición y el almacenamiento de la información. Esta situación ha complicado este análisis y la asociación (matching) de cada dato con un objeto del ECGH, sin embargo, como resultado de este trabajo se puede asegurar que los datos que se ingresarán a la BDGH, son datos útiles, fiables y certeros, de igual forma permite descubrir qué datos existen y cuáles no en el mundo real.

Las siguientes publicaciones son el resultado del trabajo realizado:

- Lecture Notes in Computer Science, 2011, Volume 6520/2011, 306-330, DOI: 10.1007/978-3-642-17505-3_14, Model-Based Engineering Applied to the Interpretation of the Human Genome [17].
- RCIS 2010 Research Challenges in Information Science, Enforcing Conceptual Modeling to Improve the Understanding of Human Genome [18].

Bibliografía

[1] **A systematic approach to modeling, capturing, and disseminating proteomics experimental data**, Chris F. Taylor, Norman W. Paton, Kevin L. Garwood, Paul D. Kirby, David A. Stead, Zhikang Yin, Eric W. Deutsch, Laura Selway, Janet Walker, Isabel Riba-Garcia, Shabaz Mohammed, Michael J. Deery, Julie A. Howard, Tom Dunkley, Ruedi Aebersold, Douglas B. Kell, Kathryn S. Lilley, Peter Roepstorff, John R. Yates III, Andy Brass, Alistair J.P. Brown, Phil Cash, Simon J. Gaskell, Simon J. Hubbard & Stephen G. Oliver. Nature Biotechnology 21, 247 - 254 (2003) doi:10.1038/nbt0303-247

[2] **Atlas – a data warehouse for integrative bioinformatics**, Sohrab P Shah, Yong Huang, Tao Xu, Macaire MS Yuen, John Ling and BF Francis Ouellette. BMC Bioinformatics 2005, 6:34

[3] **GIMS: an integrated data storage and analysis environment for genomic and functional data**, Michael Cornell, Norman W. Paton Cornelia Hedeler, Paul Kirby, Daniela Delneri,Andrew Hayes y Stephen G. Oliver. Yeast 2003; 20: 1291–1306.

[4] **Conceptual modeling of genomic information**, Paton, W.N., Khan, S., Hayes A., Moussouni, F., Brass, A., Eilbeck, K., Globe, C., Hubbard, S., Oliver, S.:. Bioinformatics. 16, 6, 548–57 (2000).

[5] **Esquema Conceptual Del Genoma Humano, Una Herramienta Para La Integración Y Gestión De Su Información**, Virrueta

G., Aremy, Departamento de Sistemas De Información y Computación, Universidad Politécnica de Valencia. (2009).

[6] **A Relational Model of Data for Large Shared Data Banks**, Codd E.F, Communications of the ACM, Volume 13 Issue 6, June 1970

[7] **Entrez Gene: gene-centered information at NCBI,** Donna Maglott, Jim Ostell, Kim D. Pruitt and Tatiana Tatusova, National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Room 5AS.13B, 45 Center Drive, Bethesda, MD 20892-6510, USA

[8] **ASN1**, http://www.itu.int/ITU-T/asn1/

[9] **The Human Gene Mutation Database** [http://www.hgmd.org]

[10] **The Human Gene Mutation Database: 2008 update**, Peter D Stenson, Matthew Mort, Edward V Ball, Katy Howells, Andrew D Phillips, Nick ST Thomas and David N Cooper, Genome Medicine 2009, 1:13

[11] http://www.ncbi.nlm.nih.gov/Sitemap/index.html

[12] **Database modeling and design**.Teorey Toby J. 1999

[13] http://www.pros.upv.es/index.php

[14] **Gestión de mutaciones en ambientes genómicos: una perspectiva basada en Modelos Conceptuales,** Burriel Coll, Veronica, Departamento de Sistemas De Información y Computación, Universidad Politécnica de Valencia. (2010).

[15] **Diseño e implementación de un entorno de carga de datos genómicos para el gen NF1 centrado en esquemas conceptuales,** Lereu Ramírez Ignacio, Departamento de Sistemas De Información y Computación, Universidad Politécnica de Valencia. (2010).

[16] **Recuperación y Procesamiento de Datos Genómicos para la Carga de Información Clasificada en una Base de Batos Basada en un Modelo Conceptual del Genoma Humano,** Rodríguez Pliego José Luis, Centro Nacional de Investigación y Desarrollo Tecnológico, Mexico (2010).

[17] **Model-Based Engineering Applied to the Interpretation of the Human Genome**, Lecture Notes in Computer Science, 2011, Volume 6520/2011, 306-330, DOI: 10.1007/978-3-642-17505-3_14, Oscar Pastor, Ana M. Levin, Matilde Celma, Juan Carlos Casamayor, Aremy Virrueta and Luis E. Eraso.

[18] **Enforcing Conceptual Modeling to Improve the Understanding of Human Genome,** RCIS 2010 Research Challenges in Information Science, Oscar Pastor, Ana M. Levin, Matilde Celma, Juan Carlos Casamayor, Aremy Virrueta and Luis E. Eraso.