Universidad de las Ciencias Informáticas

Facultad 4

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Titulo: Arquitectura de datos del subsistema Capital Humano del sistema Cedrux.

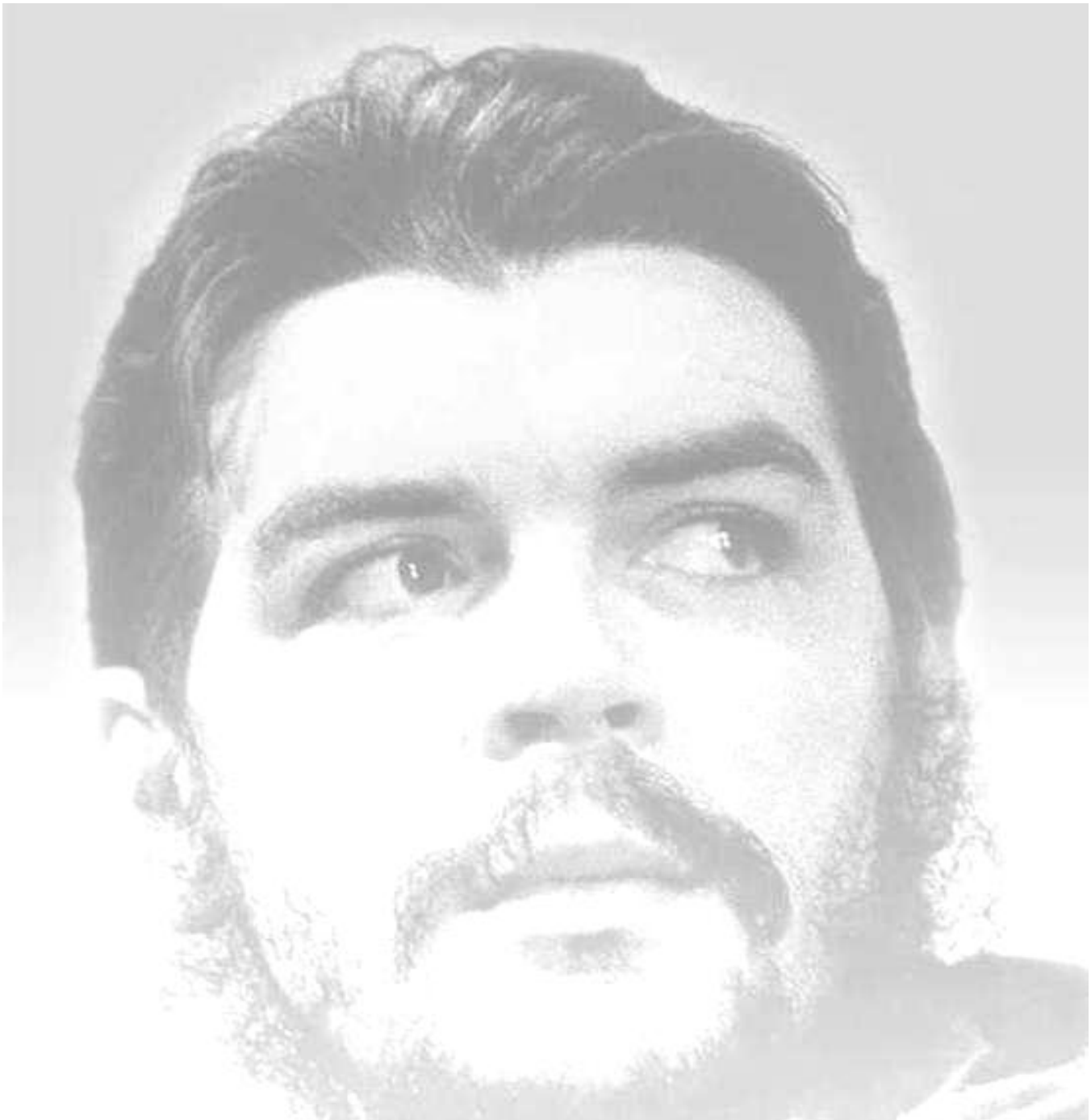
Autor: Alexei Olivero Márquez

Tutor(es): Ing. María Antonia Lajús Marrero

Ing. Alain Fernández Deronceré.

Cotutor: Ing. Javier Heredia Ruiz.

La Habana, junio de 2009



"Seamos realistas y hagamos lo imposible"
Ernesto Che Guevara
"Los obstáculos son esas cosas que las personas ven cuando dejan de mirar sus metas"
Joseph Cossman

AGRADECIMIENTOS

Jajá. Que me iba a imaginar que esta seria la parte más difícil de todo el trabajo, "Agradecimientos", una solitaria palabra que involucra a tantas y tantas personas que de una forma u otra aportaron su granito de arena y me ayudaron a lo largo de la carrera. Muchos serían los nombres que estarían en la lista, pero de manera involuntaria se me pudiese quedar alguien, por ello y de forma general, mis más sinceros agradecimientos a todos por su apoyo y de manera súper especial:

A mis padres, por su dedicación, preocupación y amor, por depositar en mi toda su confianza, por brindarme su apoyo incondicional, por ser guías y ejemplo.

A mi familia, quienes siempre han estado presente en cada momento importante de mi vida brindándome su apoyo.

A mi novia Yaneris, por haber sido mi fuente de inspiración y amor, por su amor incondicional y por depositar en mi toda su confianza.

A mis compañeros de aula y de estudio durante los cinco años.

A mis tutores, por hacer posible la realización de este trabajo, en especial a María por todo el trabajo que pasó conmigo.

A todas las personas que me apoyaron e hicieron posible que el trabajo se realizara de manera exitosa.

A Yanoski, José Luis, Richard, Edwing, Leisniel, Ismel, mis compañeros, mis amigos, mis hermanos…

A nuestro comandante en jefe Fidel Castro Ruz, forjador de esta grandiosa Revolución.

A todos los que de una manera u otra han tenido que ver con este trabajo…

A todos, Muchas Gracias.

DEDICATORIA

A mis padres, Deisy y Tito por guiarme por el difícil camino de la vida y darme todo el amor y comprensión que he necesitado.

A mis abuelos, en especial a mi abuelita Rosa, que aunque no pudo verme graduado, se que estaría muy orgullosa de mi.

A toda mi familia, que siempre me ha dado todos los buenos consejos que he necesitado y me han ayudado en todo, en especial a mi tía Nori y mi tío Felipe que no son los únicos, pero si los mas cercanos.

Y no por ultima menos especial, a mi novia Yaneris por estar siempre a mi lado y brindarme todo su cariño y comprensión, y sobre todo por esperarme todo este tiempo.

RESUMEN

En la actualidad las instituciones y empresas cubanas, han incrementado el uso de aplicaciones informáticas con el objetivo de elevar el nivel económico y tener un control eficiente sobre los recursos que manejan. Entre las aplicaciones informáticas puestas en práctica se encuentran los sistemas de gestión, que no son más que un software con el objetivo de integrar y automatizar todos los procesos que se llevan a cabo en las empresas. Estos sistemas deben contar con una arquitectura de software adecuada y por ende con la arquitectura de datos que permita persistir de manera eficiente la información que se almacena, constituyendo así, la base para el desarrollo de este tipo de sistemas.

Actualmente la Universidad de las Ciencias Informáticas se encuentra inmersa en la realización del sistema integral de gestión; Cedrux, donde se inserta el propósito de este trabajo, que persigue lograr una arquitectura de datos que posibilite soportar los datos generados por el subsistema Capital Humano.

Para lograrlo se dividió el trabajo en tres partes fundamentales. Primeramente, se realiza la fundamentación teórica, efectuando un estudio de las principales herramientas de diseño e implementación de las bases de datos y sobre los diferentes sistemas relacionados con la gestión de recursos humanos existentes a nivel mundial y que se utilizan en nuestro país. Posteriormente, se plantean elementos necesarios a tener en cuenta en el desarrollo de una arquitectura de datos, estableciendo una vista de datos; que estandariza todos los elementos a tener en cuenta durante el almacenamiento y procesamiento de los datos, además se crea el modelo de datos en el que quedan representados todos los procesos que se relacionan con la gestión de personas, trabajadores, nóminas, submayores, pagos adicionales, puestos de trabajo e incidencias. Finalmente, se analizan los resultados del trabajo y de las pruebas realizadas a la base de datos, con el objetivo de arribar a conclusiones sobre este tema y dar cumplimiento a los objetivos trazados.

PALABRAS CLAVE

ERP, Capital Humano, Arquitectura de datos, Modelo de datos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

En la actualidad, con el desarrollo tecnológico alcanzado a nivel mundial, debido al incremento en el empleo de la tecnología en casi todas las esferas de la sociedad y con el objetivo de elevar el nivel económico, muchas organizaciones, instituciones y empresas han puesto en práctica aplicaciones informáticas que le ayuden a tener un control eficiente sobre los recursos que manejan.

Las aplicaciones de este tipo deben lograr tener la arquitectura de software (AS) más adecuada, ya que es, unos de los aspectos a tener en cuenta en el proceso de desarrollo de un software. La AS establece los fundamentos para que los desarrolladores trabajen en una línea común, con el objetivo de garantizar un producto confiable que cumpla las expectativas por las cuales es desarrollado.

Dentro de la AS se encuentra la Arquitectura de Datos (AD), que se encarga de la gestión, el manejo y control de los datos; aspecto de vital importancia durante el ciclo de desarrollo de un proyecto. Tiene entre sus objetivos el modelado y el diseño de las base de datos (BD), llevar a cabo el proceso de normalización y des-normalización; todo ello con el fin de lograr un rendimiento eficiente en sistemas que manejen una gran cantidad de datos, como los sistemas de gestión de recursos o *Enterprise Resource Planning* (ERP).

Un sistema ERP es un software de gestión que tiene como objetivo integrar y automatizar todos los procesos que se llevan a cabo en las empresas. Está compuesto por diferentes módulos entre los que se encuentran Recursos Humanos, Ventas, Contabilidad y Finanzas, Compras, Producción entre otros; brindando información cruzada e integrada de todos los procesos del negocio, por lo que deben ser configurables para responder a las necesidades específicas de cada organización.

En Cuba nunca antes se había trabajado en el desarrollo de un sistema de este tipo, solamente se han adquirido algunas soluciones de gestión a un elevado precio, en las que no están integradas todas las funcionalidades que este tipo de soluciones proporciona. Por el alto valor de estos sistemas y la necesidad de su uso, fueron desarrolladas una serie de soluciones nacionales, que solo dan respuesta a una parte de las necesidades en las empresas.

La Universidad de las Ciencias Informáticas (UCI) como infraestructura productiva dedicada a la producción y desarrollo de software, tampoco había acogido la realización de un sistema como el

mencionado anteriormente. De esta manera, actualmente se lleva a cabo la realización del sistema integral de gestión Cedrux; el primero de su tipo en Cuba, que cuenta con diferentes subsistemas.

Dentro de los subsistemas se encuentra el de Capital Humano (CH), que se encarga de planificar, organizar, controlar, evaluar el desempeño y logros de los trabajadores. El subsistema CH cuenta con un módulo de Remuneración y Nómina que permite contabilizar las nóminas para el pago del personal, permitiendo configurar los tipos de impuestos y los tipos de retenciones. Garantiza la actualización del submayor de vacaciones y el de retenciones, así como la confección y emisión de comprobantes de operaciones, facilitando al personal responsabilizado llevar el control de los salarios devengados por los trabajadores en los diferentes períodos de pagos.

Dicho subsistema se encuentra en proceso de desarrollo y se ha podido apreciar que al no estar definida una AD en el mismo, imposibilita persistir los datos de una forma óptima y coherente, pudiendo llegar a ser incompletos, duplicados o erróneos; trayendo como consecuencia la toma de decisiones inadecuadas. Por otra parte, no existe un estándar a seguir con el objetivo de lograr la igualdad en el momento de definir los diferentes elementos a utilizar para el trabajo, lo que provoca que se haga lento el proceso de desarrollo, ya que los implicados tienen que recurrir a su experiencia personal o a ciertas normas que de forma tradicional se han venido aplicando.

Por todos los elementos que se han descrito y los problemas confrontados en el trabajo diario, se identificó como **problema:** ¿Cómo soportar los datos generados por el subsistema Capital Humano, de forma óptima y coherente, integrado a la solución de gestión, Cedrux?

Teniendo en cuenta el problema planteado se especifica como:

Objetivo general: Desarrollar la arquitectura de datos del subsistema de Capital Humano integrado al sistema Cedrux.

Objeto de estudio: La arquitectura de Software.

Campo de acción: La Arquitectura de datos.

Objetivos específicos:

- Realizar un estudio del estado del arte de los principales enfoques a cerca de la arquitectura de datos y su relación con los diferentes conceptos de bases de datos relacionales.

- Definir el modelo de datos relacional para el sistema de CH de la solución informática Cedrux
- Analizar la base de datos, a partir de la solución planteada para el subsistema de CH.

Tareas investigativas:

- Estudio de las herramientas que se van a utilizar para desarrollar la solución.
- Análisis de los elementos fundamentales para lograr mayor rendimiento y una base de datos optimizada.
- Estudio de los diferentes elementos que contempla la arquitectura de software.
- Selección de los principales elementos para definir una arquitectura de datos, de un sistema de gestión de CH.
- Creación de la vista de datos de la arquitectura para el subsistema de CH.
- Análisis de los diferentes elementos que se encuentran en el modelo de diseño y que representan la entrada al modelo de datos.
- Análisis y discusión de los resultados obtenidos, después de la aplicación de esta solución.

Posibles resultados: Obtención de una arquitectura de datos eficiente para el subsistema de Capital Humano integrado al sistema Cedrux.

CAPITULO 1: Fundamentación Teórica

Introducción

En el presente capitulo se realiza un estudio de diferentes temas que son de vital importancia para el desarrollo de la investigación. Se argumenta sobre las BD por el impacto que han tenido y su uso en casi todas las esferas de la sociedad. Se abordan además temas relacionados con los SGBD existentes, ya que son las herramientas de administración y gestión de las BD. También se analizan los elementos relacionados con el diseño de las mismas y las principales herramientas *CASE*. Con el objetivo de sentar las bases para su futuro uso, se abordan temas sobre los modelos de datos, con el objetivo de lograr un buen diseño de la BD, mencionándose además cuestiones relacionadas con el rendimiento y optimización de las BD. Finalmente, para un mejor entendimiento del problema a resolver, es necesario conocer que la AD indica, donde están los datos en un sistema y como están organizados, con el objetivo de controlar y cumplir obligaciones funcionales, a partir de la cual sea factible la gestión organizada de los datos que representan los distintos conceptos del negocio involucrados en un proceso.

1.1 Base de datos

Las BD tienen un impacto decisivo en el uso creciente de la computación, estas desempeñan un papel crucial en casi todas las áreas donde se aplica la computación, como son: los negocios, la ingeniería, la medicina, el derecho, le educación, la bibliotecología, el deporte, entre otros.

Entre las definiciones planteadas por algunos autores se han seleccionado:

- Conjunto de datos interrelacionados entre sí, almacenados con carácter más o menos permanente en la computadora. O sea, que una BD puede considerarse una colección de datos variables en el tiempo. (1)
- Una BD esta constituida por cierto conjunto de datos persistentes utilizado por los sistemas de aplicaciones de una empresa determinada. Un sistema de BD es un sistema computarizado cuyo propósito general es mantener información y hacer que esté disponible cuando se solicite. (2)

Una definición más rigurosa y con la que concuerda totalmente el autor es la siguiente:

- Se define como un conjunto de datos coherentes, organizados y relacionados entre sí que representan algún aspecto del mundo real, que se diseña, construye y puebla con datos específicos dirigidos a un grupo de usuarios.

1.1.1 Características de las bases de datos

Entre las principales características de los sistemas de BD se pueden mencionar: [3]

- Independencia lógica y física de los datos.
- Redundancia mínima o nula.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoria.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Ventajas de las Bases de datos [3]

- Control sobre la redundancia de datos.
- Consistencia en los datos.
- Compartición de datos.
- Mantenimiento de estándares.
- Mejora de integridad de los datos.
- Mejora en la seguridad.
- Mejora en la accesibilidad de los datos.
- Mejoras en la productividad.
- Mejora el mantenimiento.
- Aumento de la concurrencia.

Desventajas de las Bases de datos [3]

- Complejidad.
- Coste de equipamiento adicional.

- Vulnerable a los fallos.

1.1.2 Clasificación de Bases de datos

Las BD pueden clasificarse de varias maneras: [4]

- **Según la variabilidad de los datos almacenados**
  - **Bases de datos estáticas:** BD de solo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se puedan utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.
  - **Bases de datos dinámicas:** BD donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta.
- **Según el contenido**
  - **Bases de datos bibliográficas:** Solo contienen un representante de la fuente primaria, que permite localizarla. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque sino se estaría en presencia de una BD a texto completo (o de fuentes primarias). Como su nombre lo indica, el contenido son cifras o números.
  - **Bases de datos de texto completo:** Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científicas.
  - **Directorios:** Un ejemplo son las guías telefónicas en formato electrónico.

Las BD sin dudas llegaron para quedarse en nuestra sociedad, son la base potencial de todo software y ocupan un lugar determinante en cualquier área del quehacer humano, comercial y tecnológico. Brindan la posibilidad de poseer información confiable y segura en cualquier momento, y sobre todo a la hora de la toma de decisiones. Pero para que estas cumplan al máximo con la función por las cuales son desarrollas, deben de poseer un buen diseño el cual se logra realizando un buen modelo de datos (MD).

1.2 Modelos de datos

De manera general, los modelos, se usan en todas las esferas de la sociedad. Su función fundamental es simular una parte del mundo real de forma tal que sea más factible entender un problema dado, con el objetivo de intentar reproducir las características de una realidad específica.

Los MD, son el mecanismo formal para representar los datos de manera general y sistemática, establece una relación entre el mundo real y la información almacenada físicamente en la BD. (1)

Entonces, un MD no es más que la representación de un fenómeno de la realidad objetiva a través de objetos, sus propiedades y las relaciones que se establecen entre ellos.

1.2.1 Clasificación de los modelos de datos

En el proceso de abstracción que conduce a la creación de una BD, como se ha demostrado el MD desempeña una función fundamental, ya que este es utilizado para la representación de las entidades y sus características.

El MD, como contemplación del universo, es el enfoque utilizado para la representación de las entidades y sus características dentro de la BD y puede ser dividido en tres grandes tipos: (5)

- **Modelos lógicos basados en objetos:** Son aquellos que representan los datos en la forma que son captados del mundo real. El más utilizado por su facilidad de uso es el modelo Entidad-Relación (E-R).

  Se basa en una apreciación del mundo real compuesta por entidades, objetos y las relaciones entre ellos, las entidades se diferencian unas de otras a través de los atributos. Ejemplo: Una persona se diferencia de otra por sus atributos o características, como es el caso especial de su número de carnet de identidad, además de sus particularidades como nombre, apellidos, edad, estatura, etc.
- **Modelos lógicos basados en registros:** Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre ellos. Se usan para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de implementación.

  Estos modelos se dividen en:
  - **Modelo de datos jerárquicos**: Almacena su información en una estructura jerárquica. Se organiza en una forma similar a un árbol, donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y los nodos que no tienen hijos se les conocen como hojas. Es especialmente útil en el caso de aplicaciones que manejan

un gran volumen de información y datos muy compartidos, permitiendo crear estructuras estables y de gran rendimiento.

- **Modelo de datos de red:** Es ligeramente distinto, pues en este modelo un hijo puede tener varios padres; las entidades se representan como nodos y sus relaciones son las líneas que los unen, en esta estructura cualquier componente puede relacionarse con cualquier otro.
- **Modelo de datos relacional:** Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Se basa fundamentalmente, en que ofrece un sistema simple y eficaz para representar y manipular los datos, además de constituir un modelo relacional con sólidas bases teóricas.

  Modelo de datos orientado a objeto: Poseen un poder de expresión similar al de los modelos semánticos, pero los rebasan, en el sentido de que modelan explícitamente el comportamiento. Se asegura que son más fáciles de implementar y usar cuando se trata de crear aplicaciones. Es bastante reciente y propio de los modelos informáticos orientados a objetos.

1.2.2 Modelo entidad relación

Se trata de un modelo que sirve para crear esquemas conceptuales de BD. Es prácticamente un estándar para la realización de estos. Toma como punto de partida considerar la existencia de entidades, que representan objetos, personas, etc., sobre las que se quiere almacenar información relevante.

Las entidades con las mismas características forman un tipo de entidad. A las características necesarias para describir completamente a cada tipo de entidad se les denominará atributo. Posteriormente, las entidades y sus atributos se representan físicamente a través de tablas (transformación en un modelo relacional) en las que los datos se almacenan en dos dimensiones. Las filas de la tabla contienen los atributos de cada una de las entidades, y las columnas el conjunto de atributos del mismo tipo de cada entidad. El grado de la tabla corresponderá al número de columnas de la tabla.

Las entidades pueden ser de dos tipos:

- **Entidad regular:** en la que se puede definir la clave primaria dentro de sus propios atributos.

- **Entidad débil:** aquellas que no pueden utilizar sus propios atributos como clave, al estar asociada a otra entidad.

Uno de los puntos fuertes de este modelo es que prevé que las entidades puedan mantener relaciones entre ellas. Las relaciones entre tablas, basadas en la conexión de éstas a través de las claves, pueden ofrecer diferentes cardinalidades, estas pueden ser:

- **Cardinalidad mínima:** Indica el número mínimo de asociaciones en las que aparecerá cada ejemplar de la entidad (el valor que se anota es de cero o uno).
- **Cardinalidad máxima:** Indica el número máximo de relaciones en las que puede aparecer cada ejemplar de la entidad.

Los MD son la base de un buen diseño. Los elementos fundamentales de este modelo, se encuentran inmersos en las entidades, los atributos y sus relaciones. Permite agrupar los datos en forma de objetos y relacionarlo con otras entidades.

1.3 Herramientas para el diseño de bases de datos

Las herramientas *Computer Aided Software Engineering* (*CASE*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el costo de los mismos en términos de tiempo y dinero.

Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en el aumento de la productividad en las áreas de desarrollo y mantenimiento de los sistemas informáticos, mejorar la calidad del software desarrollado, reducir en tiempo y costo el desarrollo y mantenimiento del software, además de que permiten desarrollar el diseño de los modelos conceptuales y lógicos, para la generación del MD.

La introducción de las herramientas *CASE* para ayudar en este proceso, ha permitido que los diagramas puedan ser fácilmente creados y modificados, mejorando la calidad del diseño de software. Los diccionarios de datos, un documento muy usado que mantiene los detalles de cada tipo de dato y los procesos dentro de un sistema, son el resultado directo de la llegada del diseño de flujo de datos y análisis estructural, hecho posible a través de las mejoras en las herramientas *CASE*. [(6)]

1.3.1 Clasificación de las herramientas CASE

No existe una única clasificación para este tipo de herramientas y en ocasiones, es difícil incluirlas en una clase en común. Estas podrían clasificarse de la forma siguiente: (7)

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que abarca.
- La arquitectura de las aplicaciones que produce.
- Su funcionalidad.

Las herramientas *CASE*, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

- **Herramientas integradas:** Abarcan todas las fases del ciclo de vida del desarrollo de sistemas.
- **Herramientas de alto nivel:** Orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.
- **Herramientas de bajo nivel:** Dirigidas a las últimas fases del desarrollo: construcción e implantación.
- **Juegos de Herramientas:** Son el tipo más simple de herramientas *CASE*. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento. (6)

1.3.2 Puntos importantes a considerar al seleccionar una herramienta CASE

Seleccionar una herramienta *CASE* no es una tarea simple. No existe una mejor herramienta respecto a otra. Hay numerosas historias respecto al uso de *CASE* y las fallas que pueden producirse. Las fallas o las respuestas satisfactorias están en relación con las expectativas. Si el proceso de evaluación y selección de la herramienta *CASE* falla, entonces la herramienta no cumplirá con las especificaciones o expectativas del negocio. Esto puede ocurrir durante el proceso de implementación o ejecución del producto.

Hay tres puntos comunes que fallan en el proceso de evaluación y selección: (6)

- **El proceso en sí mismo:**

  El proceso de evaluación y selección de Herramientas *CASE* debe aproximarse a un proyecto mayor. El proceso debe definirse cuidadosamente y debe incluir las mejores técnicas de dirección de proyecto. Ninguna selección es igual que otra, porque dos organizaciones no son iguales.

  Por ejemplo, el proceso de selección de una herramienta *CASE* para realizar un software para el Ministerio de Finanzas y Precios puede ser completamente diferente que una para realizar un software para la Aduana. Aunque hay principios básicos para la selección de las mismas, por ejemplo, se debe entender el criterio en el que está basado el proceso de selección, se debe tener una visión común.

- **Los pre-requisitos necesarios:**

  El propósito de las herramientas *CASE* es apoyar y facilitar el desarrollo de software. Debe haber una comprensión clara del propósito de las herramientas que se propongan dentro del ambiente de desarrollo que es compartido por el equipo de la selección. El equipo debe tener una visión común del ambiente de desarrollo de sistemas, resultando la selección de la herramienta adecuada.

  Otro requisito previo importante sería tener una metodología de desarrollo de sistemas seleccionada. Sin una metodología, ingresará al largo camino del fracaso. Las herramientas implementan la metodología, no la determinan.

- **Conocer la organización:**

  Cuando se evalúa y selecciona una herramienta *CASE*, es importante conocer y entender a la organización. Tal como las personas son únicas, así también las organizaciones son únicas a su propio modo, cada una tiene una personalidad e infraestructura propia. Una empresa podría disciplinarse y alcanzar un nivel alto de madurez en el proceso de diseño de software, mientras otras pueden estar en las fases tempranas. Sin tener en cuenta la disciplina y la madurez, es muy importante entender la organización que se verá reflejada en la selección final.

1.3.3 Herramientas CASE

Un buen diseño de la BD no es trivial y de esto depende el buen funcionamiento de la aplicación, por lo que es muy recomendable usar una herramienta que permita hacerlo. Actualmente hay muchas herramientas a nivel mundial que fueron desarrolladas con este fin, pero a veces son costosas y no es factible tener estos gatos.

A partir del estudio realizado se identificaron muchas herramientas que se utilizan para el diseño de BD, entre las que se encuentran:

- *DBDesigner*.
- *ER/Studio*.
- *Rational Rose*.
- *Enterprise Architect.*
- *Visual Paradigm.*

Por la importancia que representa, se decide profundizar en las características y funcionalidades de las herramientas *CASE* mencionadas anteriormente, ya que son las utilizadas para el modelado y diseño de las BD.

DBDesigner

Es un software libre diseñado y optimizado para *MySQL* que permite diseñar BD para las aplicaciones en internet o cualquier BD para aplicaciones desarrolladas en diferentes lenguajes.

Principales características: [8]

- Modelados realizados en XML.
- Interfaz de usuario amigable:
- Permite hacer ingeniería inversa desde BD como *MySQL*, *Oracle*, *Microsoft SQL Server* y cualquier *Base de Datos ODBC*.
- Sincronización automática con BD *MySQL*. Esta función es una de las más interesantes porque permite actualizar las tablas y relaciones existentes en una BD *MySQL* que haya instalada en un ordenador local o remoto, sin necesidad de ejecutar ningún script.
- Colocación automática de las ***foreign key***.

- Disponible para *Linux* y *Windows*.
- Es software libre y licenciado bajo la *GNU GPL*. Esto significa que se pueden descargar ejecutables así como el código fuente del programa y usarlo de forma gratuita.

ER/Studio.

Es una herramienta de modelado de datos fácil de usar y multinivel para el diseño de BD a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de BD, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejas.

Está equipado para crear y manejar diseños de BD funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes. (9)

Es simple y fácil de manejar, ayuda a organizaciones para tomar decisiones, eliminar redundancia y alcanzar en última instancia usos de más alta calidad que entreguen datos más eficientes y exactos.

Entre las características que posee se pueden destacar: (9)

- Sincronización de las BD. Sincronización entre el modelo físico y el lógico. Mezcla entre cualquier par de diagramas físicos para la misma plataforma de bases de datos. Compara lado a lado las diferencias y da la opción al usuario para decidir las diferencias a mezclar o ignorar.
- Objetos reusables, construyendo atributos reusables y aplicarlos a atributos y columnas.
- Crear rápidamente sub-vistas y sub-modelos eligiendo un área del diagrama.
- Creación de un MD ofreciendo la posibilidad de crear un MD nuevo, realizar ingeniería inversa a partir de una base de datos ya existente o importar un archivo *SQL* o MD realizado con *Edwin*.
- Publicación automática en la Web. Puede documentar automáticamente un diagrama entero, generando un conjunto integrado de reportes HTML sofisticados que múltiples usuarios pueden compartir en Internet
- Calidad de presentación en los reportes. Además de los reportes en HTML, puede generar reportes de alta calidad con un formato de texto amplio que está disponible para presentaciones profesionales.
- Es software propietario.

Rational Rose

Rational Rose es una herramienta *CASE* propietaria que soporta el modelado visual con UML, ofreciendo distintas perspectivas del sistema. Da soporte al Proceso Unificado de Rational (RUP). Algunas de sus características son: (10)

- Diseño centrado en casos de uso y enfocado al negocio, que generan un software de mayor calidad.
- Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.
- Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.
- Desarrollo Iterativo
- Generador de Código
- Ingeniería Inversa

Enterprise Architect

Enterprise Architect es una herramienta CASE de diseño y análisis UML. Cube el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Diseñada para ayudar a construir software robusto y fácil de mantener.

Entre sus características fundamentales se encuentran: (11)

- Es software propietario
- Es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento.
- Provee soporte para pruebas, mantenimiento y control de cambio.
- Diseña y genera elementos de BD.
- Realiza ingeniería inversa para sistemas de BD muy populares como *Oracle*, *SQL Server*, *MySQL*, *Access*, *PostgreSQL*, etc.

- Permite generar tablas del Modelo de BD, columnas, claves, llaves foráneas, etc.
- Permite generar los scripts DLL para crear las estructuras de BD.
- Posee gran velocidad, estabilidad y buen rendimiento.

Visual Paradigm

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software, análisis y diseño orientado a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es propietario y multilenguaje.

Principales características de *Visual Paradigm*: [12]

- Ingeniería ida y vuelta.
- Ingeniería inversa.
- Diagramas de flujo de datos.
- Generación de BD.
- Transformación de diagramas ER en tablas en la BD.
- Ingeniería inversa en BD, desde SGBD existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.

Por las características mencionadas anteriormente fue elegida por la dirección del proyecto para ser la herramienta que guiara el proceso de diseño y construcción del sistema Cedrux.

Sin lugar a dudas las herramientas *CASE* han venido a revolucionar la forma de automatizar los aspectos clave en el desarrollo de los sistemas de información, debido a la gran plataforma de seguridad que ofrecen a los sistemas que las usan, brindando toda una gama de componentes que incluyen todos o la mayoría de los requisitos necesarios para el desarrollo de los sistemas. Además les brinda a los Arquitectos de datos un arma potencial ya que les posibilita la generación de script o la generación directa de la BD hacia el SGBD que se vaya a utilizar.

1.4 Sistemas gestores de bases de datos

Un SGBD es un software que permite introducir, organizar, recuperar información y administrar las BD. El objetivo principal de los SGBD es proporcionar un entorno que sea a la vez conveniente y eficiente al ser utilizado para extraer y almacenar información en la BD, otros de sus objetivos son suministrar la interfaz entre el conjunto de datos y los usuarios.

El SGBD es un conjunto integrado de programas, procedimientos, lenguajes, etc., que suministra, tanto a usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos, manteniendo su integridad, confidencialidad y seguridad.

Ventajas de los SGBD

- Proveen facilidades para la manipulación de grandes volúmenes de datos.
- Simplifican la programación de chequeos de consistencia.
- Permiten realizar modificaciones en la organización de los datos con un impacto mínimo en el código de los programas.
- Permiten implementar un manejo centralizado de la seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
- Tiempos de desarrollo bajos y aumento de la calidad del sistema desarrollado.
- Suministran interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

Desventajas de los SGBD

- Es necesario disponer de una o más personas que administren de la base de datos.
- Son software muy complejo y las personas que vayan a usarlo deben tener conocimiento de las funcionalidades del mismo para poder aprovecharlo al máximo.
- La complejidad y la gran cantidad de funciones que tienen hacen que sea un software de gran tamaño, que requiere de gran cantidad de memoria para poder correr.
- Coste del hardware adicional, los requisitos de hardware para correr un SGBD por lo general son relativamente altos, por lo que estos equipos pueden llegar a costar gran cantidad de dinero.

1.4.1 Análisis de diferentes SGBD

Hoy en día existe gran cantidad de gestores, cada uno con características distintas, así como ventajas y desventajas a la hora de ser usado. A continuación se presenta un listado con algunos de los diferentes SGBD existentes divididos en tres grupos:

- **SGBD libres.**
  - *MySQL.*
  - *Firebird.*
  - *SQLite.*
  - *PostgreSQL*.
- **SGBD gratuitos.**
  - *Microsoft SQL Server Compact Edición Básica.*
  - *Sybase ASE*.
- **SGBD comerciales.**
  - *Advantage Database.*
  - *DBase.*
  - *Fox Pro.*
  - *IBM Informix.*
  - *Microsoft Access.*
  - *Microsoft SQL Server.*
  - *Oracle*.
  - Otros.

A continuación se describen las características de algunos de los SGBD libres mencionados con anterioridad.

MySQL

Es uno de los sistemas de gestión de BD más populares y utilizados entre los desarrolladores de aplicaciones web. Su gran ventaja frente a otros sistemas de gestión como Oracle o SQL Server, es que se ofrece como *GNU GPL*. Hoy día existen multitud de *APIS* que permiten conectar prácticamente cualquier lenguaje de programación con esta BD: como por ejemplo *C*, *C++, C#*, *Pascal*, *Delphi*, *Java*, *PHP* etc. [(8)]

Entre sus características más significativas tenemos: [13]

- Alto rendimiento.
- Bajo costo.
- Facilidad de configuración y aprendizaje.
- Portabilidad.
- Accesibilidad al código fuente.

Firebird

Es *open source* y no hay licencias duales. Tanto para uso comercial como para aplicaciones *open source*, es totalmente libre. La tecnología de *Firebird* lleva 20 años funcionando, esto hace que sea un producto muy maduro y estable. Entre sus características fundamentales se encuentran: [14]

- Completo soporte para Procedimientos Almacenados y Disparadores.
- Integridad Referencial.
- Bajo consumo de recursos.
- Completo lenguaje interno para procedimientos almacenados y disparadores (*PSQL*).
- Copias de seguridad incrementales.
- Tablas Temporales.

SQLite

Es una pequeña librería libre programada en lenguaje C que implementa un completo motor de BD que no precisa configuración, es una BD muy rápida y su ventaja fundamental es que permite utilizar el lenguaje estándar *SQL* además de su versatilidad. Entre sus principales características se encuentran: [15]

- Cero configuraciones o administración necesaria.
- Una BD completa es almacenada en un solo archivo.
- Soporta tipos de datos *BLOB*.
- Simple y fácil de usar los *API* de desarrollo.

Así como cuenta de algunas ventajas y características realmente asombrosas, también cuenta con algunas limitaciones como son: [16]

- **Limitaciones en Where:** esta limitación está dada por el soporte para clausuras anidadas.
- **Falta de Clave Foránea:** se hace caso omiso de las claves foráneas; esto quiere decir, cuando se realice la creación de la tabla desde el modo consola, está permitiendo el uso de la clausura, aunque no realizara el chequeo de la misma.
- **Soporte completo a TRIGGERS:** No se pueden programar funciones disparadoras.

PostgreSQL

PostgreSQL es el trabajo colectivo de cientos de desarrolladores, basándose en veintiún años de desarrollo que empezaron en la Universidad de California en *Berkeley*. Con su soporte de larga data de un conjunto de características de nivel empresarial, el cual incluye transacciones, funciones, ***triggers*** y sub-consultas. *PostgreSQL* está siendo usado por muchas de las más exigentes empresas y agencias de gobierno. Se distribuye bajo licencia BSD[1] (*Berkeley Software Distribution)*, lo cual permite el uso y distribución sin costo, tanto para aplicaciones comerciales como no comerciales. (17)

PostgreSQL está ampliamente considerado como el sistema de BD de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre. Estas características son: (18)

- **Integridad referencial:** Es utilizada para garantizar la validez de los datos de la BD.
- **Bloqueo de tablas y filas:** Ofrece varios modos de bloqueo para controlar el acceso concurrente a las tablas. Algunos de estos modos de bloqueos los adquiere automáticamente antes de la declaración de una declaración, mientras que otras son proporcionadas para ser usadas por las aplicaciones.
- **Constraints y trigger:** Tienen la funcionalidad de mantener la integridad y la consistencia en la BD.
- **Múltiples tipos de datos predefinidos:** Como todos los manejadores de bases de datos, Pos*t*greSQL implementa los tipos de datos definidos para el estándar *SQL3* y aumenta algunos otros.

[1] BSD: Pertenece al grupo de licencias de software libre, BSD al contrario que la GPL permite el uso del código fuente en software no libre.

- **Soporte de tipos y funciones de usuario:** Soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. Incorpora una estructura de datos *Array*.

A partir de las funcionalidades que ofrece, reflejadas en cada uno de los elementos dados; que lo convierten en un potente gestor, es escogido por el equipo de dirección del proyecto para que fuese el SGBD a utilizar en el desarrollo del proyecto. Por otro lado, es de los gestores libres el más probado en la UCI por la comunidad de desarrollo. Además, garantiza los requisitos necesarios del proyecto; como clúster, balance de carga, procedimientos almacenados, vistas, herencia y **triggers**.

Los SGBD han tenido un gran desarrollo debido a su importancia y uso en el proceso de desarrollo de diferentes e importantes proyectos. Esto se debe porque permiten disponer de los datos fácilmente, logrando que estos sean accesibles, entendibles y utilizables. Además de que ofrecen la posibilidad de administración y control sobre las BD, permitiendo que se logre un rendimiento adecuado de la misma.

1.5 Rendimiento y optimización de bases de datos

Uno de los aspectos fundamentales a tener presente en buena parte de las etapas de diseño y desarrollo de una BD, es el rendimiento futuro de la misma. Desde el momento mismo en que se elige una configuración de hardware y herramientas de software para el desarrollo, los impactos sobre el rendimiento deben evaluarse cada vez que se tome una decisión de diseño, y en última instancia, durante toda la fase de configuración.

El rendimiento es muy importante en cualquier sistema de información. Si se desea obtener buenos resultados, es fundamental reducir los tiempos de espera, aumentar la productividad y maximizar el rendimiento de la BD.

Cuando se diseña una BD, se debe asegurar la realización de todas las operaciones importantes de forma rápida y correcta. Algunos problemas de rendimiento se pueden resolver una vez que la BD se encuentra en producción, sin embargo otros pueden ser el resultado de un diseño inadecuado y se pueden solucionar mediante el cambio de la estructura y el diseño de la BD.

Cuando se diseña e implementa una BD, deben identificarse las tablas de gran tamaño y los procesos más complejos que realizará. Además, deben considerarse los efectos que puede tener en el rendimiento el aumento del número de usuarios con acceso a la misma.

Existen varias operaciones que pueden llevarse a cabo para optimizar las BD entre las que se encuentran:

- **Diseñar tablas sin datos redundantes:** Una BD bien diseñada es un requisito básico para una rápida recuperación y actualización de los datos.
- **Seleccionar los tipos de datos apropiados para los campos:** Se puede ahorrar espacio mediante la selección de tipos de datos apropiados para los campos. Al definir un campo, seleccionar el tipo de datos o el tamaño de campo más pequeño que sea apropiado para los datos del campo.
- **Crear índices para los campos que se ordenen o combinen, o para los que se establezcan criterios:** Se puede mejorar considerablemente la velocidad de las consultas mediante la indexación de campos y la indexación de cualquier campo utilizado para establecer criterios para la consulta. Pero es importante ser cuidadoso en el momento de definirlos.

Particularmente el tema de los índices se abordará con mayor detalle más adelante, por lo que representan dentro de las BD. Sin embargo es preciso mencionar que; no es adecuado utilizar índice en todos los casos ni indiscriminadamente.

1.5.1 Proceso de mejoras y optimización de bases de datos

Una BD se diseña a nivel físico para lograr un buen rendimiento de sus aplicaciones, sin embargo el esquema físico producto de este proceso de diseño es apenas una determinación inicial de estructuras y caminos de acceso. Luego, con la operación de las aplicaciones sobre la BD, se deben medir los tiempos de respuesta de las transacciones, si estas mediciones no están dentro de los rangos esperados es necesario llevar a cabo un proceso de "entonación de la base de datos". [(19)]

Principios de Entonación

Denni Sasha plantea tres principios básicos para la entonación de BD y estas son: [(20)]

Pensar globalmente, arreglar localmente: Para ser efectivos en la tarea de entonación se debe identificar el problema adecuadamente y hacer una intervención mínima en el sistema. No basta con ver las estadísticas, ni con entonar una consulta en particular, es necesario ver el problema completo.

El inicio es costoso, continuar es más barato: Al comenzar a leer datos de un disco, la ubicación del primer bloque es lo más costoso, pues involucra ubicar ese primer bloque y mover todas las piezas a

ese lugar. Una vez ubicado el primer bloque, leer los siguientes es menos costoso, entonces debemos tratar de hacer esa ubicación inicial la menor cantidad de veces posible. El consejo en este caso es: lograr lo que hace falta llevar a cabo usando el menor número de inicios posible.

Al servidor lo que es del servidor: Decisiones de donde realizar cada tarea, en el servidor o en el programa de aplicación. Es necesario diseñar desde el principio donde se van a ejecutar las diferentes tareas relativas a la BD (el servidor) y al programa de aplicación (el cliente).

La entonación de una BD relacional, se puede dividir en dos componentes fundamentales a saber, la entonación del diseño físico y la entonación de las consultas. Con respecto al diseño físico, se define la organización primaria de los archivos que van a contener las tablas de una BD tomando en cuenta el uso que se le va a dar, esto se refiere a cuales transacciones están definidas sobre la BD y cual es su frecuencia de ejecución. También se toman en consideración las restricciones de tiempo de respuesta de cada transacción. El rendimiento de una BD se mide, entre otros aspectos, por los tiempos de respuesta de sus transacciones y por el *throughput* [2]. (19)

Por otra parte, las consultas tienen más variedad de formas y espacio para mejorar el rendimiento, por lo que conviene distinguir los tipos de consultas que se pueden escribir. La tipificación propuesta por Sasha consiste de ocho clases de consultas que a continuación se describen: (20)

- **Consulta puntual (“*point query*”):** La condición de búsqueda es igualdad y la consulta devuelve máximo una tupla.
- **Consulta múltiple (“*multipoint query*”):** La condición de búsqueda es igualdad y la consulta devuelve varias tuplas.
- **Consulta por rango (“*range query*”):** El rango se especifica sobre los valores de un atributo definido en un dominio ordenado, la consulta devuelve todas las tuplas cuyo valor del atributo de la condición está dentro del rango especificado.
- **Consulta de prefijo (“*prefix match query*”):** Se expresa sobre un atributo o secuencia de atributos, la condición de búsqueda especifica solo el prefijo del valor buscado.
- **Consulta extrema (“*extremal query*”):** Devuelve las tuplas cuyo valor en uno o más atributos es un mínimo o un máximo.

[2] Throughput: Número de transacciones por unidad de tiempo.

- **Consulta de ordenamiento ("*ordering query*"):** Es aquella consulta que incluye una cláusula de ordenamiento de las tuplas resultantes por el orden del valor de un atributo en particular.
- **Consulta de agrupación ("*grouping query*"):** Es aquella consulta cuyas tuplas resultantes se agrupan de acuerdo al valor de uno o más atributos, generalmente a cada grupo se le aplica una función agregada.
- **Consulta con Join ("*join query*"):** Es aquella consulta que asocia dos o más relaciones a través de atributos comunes. La condición sobre la cual se asocian las tablas puede usar la igualdad o algún otro comparador.

1.5.2 Índices

Como se dijo anteriormente los índices son necesarios (aunque no en todos los casos) a la hora de alcanzar un mayor rendimiento en nuestra BD. Por tal razón se darán una serie de características importantes a tener en cuenta a la hora de la utilización o no de los índices.

Índices primarios: Contiene los registros completos del archivo de datos, de modo que el orden definido por la clave del índice es el orden en el cual están almacenados los registros de datos. Por ejemplo, el archivo de datos de Estudiantes puede estar organizado por el carnet del estudiante y tener un índice primario por el carnet del estudiante.

Ventajas de los índices primarios [19]

- Si el índice primario es no denso tendrá menos apuntadores que un índice denso, lo cual puede ahorrar un acceso a disco por registro, si los registros son pequeños.
- Un índice primario es bueno para consultas múltiples.
- Un índice primario puede soportar consultas por rango, de prefijo y con ordenamiento.

Índice secundario: provee un camino de acceso adicional a los datos de un archivo cuya organización primaria no está relacionada con el ordenamiento de la clave del índice. Ejemplo el archivo de Estudiantes que está organizado por carnet, puede tener un índice secundario por el nombre del estudiante, permitiendo así un acceso a los datos relativamente directo por el nombre del estudiante.

Ventajas de los índices secundarios [19]

- Puede eliminar la necesidad de acceder el archivo de datos para algunas consultas, lo cual puede proveer un mejor rendimiento que un índice primario.
- Para tablas críticas que tienen pocos atributos puede ser conveniente crear varios índices secundarios para que todas las consultas sobre determinados atributos estén cubiertos (es decir, se puedan responder sin acceder el archivo de datos).
- El mejor uso de los índices secundarios es cuando cubren una consulta sin tener que ir al archivo de datos. Son buenos si el resultado de una consulta tiene un número significativamente menor que el número de bloques que ocupa el archivo.
- Un índice secundario siempre es útil para consultas puntuales, pero para consultas múltiples depende de los valores distintos del atributo que pueda haber.

Índices compuestos: Es aquel que está definido sobre más de un atributo. Consideremos una relación ESTUDIANTE, se puede crear un índice que combine el año de ingreso a la carrera con el código de la carrera que esta estudiando. Para cada año de ingreso este índice agruparía todos los valores de código de carrera y apuntaría en cada año a todos los estudiantes de esa carrera. Para la consulta cuantos estudiantes ingresaron a la carrera determinada en un año determinado, bastaría con acceder este índice y no habría necesidad de consultar el archivo de datos.

Ventajas de los índices compuestos [19]

- Un índice compuesto puede responder a algunas consultas sin necesidad de acceder al archivo de datos. Por ejemplo, para saber el número de estudiantes inscritos en una determinada carrera, basta con consultar el índice secundario sobre el atributo código dela carrera de la relación ESTUDIANTE.
- Una consulta sobre todos los atributos de un índice compuesto es muy posible que devuelva menos tuplas que una consulta sobre algunos de los atributos del índice compuesto.
- Un índice compuesto se puede utilizar para controlar eficientemente la unicidad de varios atributos (los que componen el índice compuesto).
- Pueden soportar mejor algunos tipos de consultas espaciales. Lo ideal para consultas espaciales es almacenar los datos en una estructura multidimensional apropiada para ello, sin embargo, hay algunos casos donde el índice compuesto puede ayudar. Por ejemplo, si para una ciudad se

almacenan atributos de latitud y longitud y se tiene un índice compuesto sobre la combinación, latitud, longitud, en ese orden, las consultas que pidan una latitud particular y un rango de longitudes, puede hacerse más rápida con un índice compuesto que sin el.

Un archivo de datos puede tener un solo índice primario, pero varios índices secundarios. Los índices primarios pueden ser densos o no densos, en algunos sistemas se obliga a que los primarios sean de una forma o de otra. Los índices secundarios solo pueden ser densos. (19)

La presencia de índices reduce la concurrencia en las aplicaciones de múltiples usuarios y reduce el rendimiento de la BD cuando se actualizan datos de campos que están indexados o cuando se agregan o eliminan registros. Al agregar un índice se puede acelerar una consulta, pero también se puede enlentecer. Por ejemplo, al agregar un índice a un campo puede obtenerse una ganancia de rendimiento muy escasa si otros campos de la tabla ya están indexados.

1.6 Sistemas que contemplan gestión de recursos humanos

En este epígrafe se realiza un estudio de las negativas existentes en los sistemas más importantes y reconocidos a nivel mundial y nacional, en los cuales se maneja la gestión de recursos humanos.

Estos sistemas de gestión de recursos humanos, tienen la responsabilidad de ayudar a dirigir y controlar la organización, así como mantener un control estricto de todos los recursos de la misma.

1.6.1 Sistemas internacionales

ASSETS NS

Es un software extranjero, lo cual significa gastar recursos monetarios en adquirirlo. Además de que para su instalación se requiere obligatoriamente el sistema operativo *Windows*, utilizando como SGBD el *Access* 97.

SAGE MAS 500

Es un software extranjero no certificado, no esta aprobado para que se utilice en Cuba. Esta desarrollado para plataforma de software propietario. Para que funcione correctamente necesita de una integración con el resto de los subsistemas, por lo que para que el funcionamiento sea el más óptimo posible debería de comprarse el sistema completo, siendo poco útil adquirir el software ya que muchos de sus módulos no se utilizarían.

SAP

Aun estando certificado para su posible utilización en Cuba, no es una solución factible al problema que se expone, debido a que el país no cuenta con los recursos financieros suficientes como para adquirirlo y sus licencias tienen un alto valor monetario. Además es un software extranjero y desarrollado para plataforma de software propietario.

Seven 2000

Es un software extranjero no certificado, no esta aprobado para que se utilice en Cuba. Esta desarrollado para plataforma de software propietario. Además no presenta una gestión de recursos humanos integrado a la nómina.

1.6.2 Sistemas nacionales

RODAS XXI

No es una solución factible debido a que no presenta una gestión de recursos humanos integrado a la nómina. Además está desarrollado para plataforma de software propietario.

VERSAT SARASOLA

No presenta una gestión de recursos humanos integrado a la nómina. Al igual que otros de los sistemas analizados anteriormente se deben comprar las licencias del mismo, viéndose en la necesidad de comprar el sistema en su totalidad para que funcione correctamente, debido a que necesita de una integración con el resto de los subsistemas. Además de estar desarrollado para software propietario.

Como se pudo observar, los sistemas dedicados a la gestión de recursos humanos mencionados anteriormente, ninguno cumple con las características necesarias para ser puestos en práctica en nuestro país. La mayoría de ellos no presentan la gestión de los recursos humanos integrados a la nómina, además están desarrollados para plataforma de software propietario y se debe pagar un alto precio por sus licencias.

Conclusiones

Con el estudio realizado a lo largo de este capitulo y evaluando cada uno de los argumentos planteados, se puede concluir que el desarrollo creciente de las BD y la importancia de estas en la

sociedad, junto a sus ventajas, han logrado que estas alcancen un punto de importante impacto a la hora de desarrollar un producto con calidad, brindando la posibilidad de almacenar una gran cantidad de información.

Ciertamente, el diseño de BD es cada vez más importante, ya que un error en el mismo o en la interpretación de los datos puede conllevar a la toma de una decisión equivocada. Para evitar esto es esencial diseñar estructuras de datos correctas y fiables, minimizando los tiempos de diseño y facilitando su comprensión. Para realizar un buen diseño se atraviesan distintas etapas, entre las que se encuentra como una de las primeras y más importantes la modelación.

Los SGBD han alcanzado un gran desarrollo en nuestra sociedad, mediante estos se logra organizar, controlar y administrar la información almacenadas en las BD. Estos proporcionan seguridad, integridad y rapidez a la hora de almacenar, utilizar y extraer información. Además brindan la posibilidad de realizar respaldo y recuperación de la información.

Se estudiaron diferentes herramientas *CASE* que se utilizan para el diseño de las BD, con el objetivo de mejorar la productividad así como minimizar el costo. Facilitan el proceso de desarrollo, haciéndolo más eficaz y mejorando eficientemente la calidad del producto. Entre sus beneficios se encuentra la potenciación en la mejora del producto final, facilitación en el desarrollo de los procesos, mejorar la calidad del sistema y la disminución de tiempo.

Se estudiaron además una serie de factores que influyen directamente en el rendimiento de las BD, entre ellos y el más importante el tema de los índices. Estos mejoran significativamente el rendimiento, pero no es adecuado utilizar índices en todos los casos ni indiscriminadamente ya que pueden llegar a ser seriamente perjudiciales.

Además se llego a la conclusión de que los sistemas existentes a nivel mundial y nacional para la gestión de los recursos humanos, no cumplen con las características necesarias para su puesta en práctica en nuestro país.

CAPITULO 2: Desarrollo de la solución

Introducción

Este capítulo constituye la parte fundamental del presente trabajo. En él se realiza la descripción de la solución propuesta para darle solución al problema a resolver. Se describe superficialmente en que consiste y los elementos que contempla la AS. Se profundiza en la definición del término AD y los aspectos que contempla. Se describen además, las clases persistentes que son el resultado de la interpretación del modelo de diseño, dando entrada así al modelo de datos. La integración entre CH y los demás componentes, realizándose la matriz de integración. Finalmente, se realiza la descripción, construcción y muestra final del MD relacional que constituye la solución, incluyendo las relaciones del mismo.

2.1 Arquitectura de Software

La AS es una disciplina reciente en el mundo del desarrollo de software, pero se ha demostrado que representa un elemento de vital importancia dentro del ciclo de desarrollo de un proyecto de software; facilita la comunicación entre todas las partes interesadas en el desarrollo de un sistema, destaca decisiones tempranas de diseño que tendrán un profundo impacto en todo el trabajo de ingeniería de software que sigue, constituye un modelo relativamente pequeño y comprensible de cómo esta estructurado el sistema y de cómo trabajan juntos sus componentes.

La AS es la organización fundamental de un sistema, formada por componentes, las relaciones entre ellos, el contexto en el que se implantarán y los principios que orientan su diseño y evolución. Es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y constituye un puente entre el requerimiento y el código.

Beneficios de la Arquitectura de Software

- Decisiones tempranas.
- Análisis de consistencia antes de elaborar el diseño y escribir el código.
- Sistematización de la experiencia.
- Homogenización del lenguaje.
- Re-utilización.

Dentro de la AS, según la arquitectura llevada a cabo en el proyecto ERP-Cuba, se encuentran una serie de vistas comprendidas, que en su conjunto conforman a esta. Entre estas vistas se encuentran, la vista de Tecnología, la vista de Integración, la vista de Sistema, la vista de Infraestructura, la vista de Presentación, la vista de Seguridad, y la vista de Datos.

La AS sin dudas es el aspecto fundamental en el proceso de construcción de un software, proporciona y agrupa los detalles completos del software que se va a construir y asegura que la construcción del mismo sea correcta y completa. Uno de los elementos fundamentales de la AS es la AD, que es la base para la creación de aplicaciones con alta calidad.

2.2 Arquitectura de datos

El diseño de datos, a veces llamado AD, crea un modelo de datos y/o información que se representa con un alto nivel de abstracción. Este MD, es entonces refinado en progresivas representaciones específicas de la implementación. En muchas aplicaciones de software, la AD tendrá una gran influencia sobre la arquitectura del software que debe procesarlo. (21)

La estructura de datos ha sido siempre una parte importante del diseño de software. Al nivel de los componentes del programa, el desafío de las estructuras de datos y de los algoritmos asociados requeridos para su manipulación, son la parte esencial en la creación de aplicaciones de alta calidad. A nivel de aplicación, la traducción de un MD en una BD es el punto clave para alcanzar los objetivos del negocio del sistema. (21)

La AD es una de las primeras de las arquitecturas a ser definidas; identifica y define las mejores clases de datos y atributos que apoyan las funciones del negocio. Tiene como principal objetivo definir el MD, identificando sus entidades, los atributos y las relaciones entre ellas. Además, tiene el propósito de documentar la estructura y la definición de la información de los datos, describiendo como estos están representados y que significan.

2.2.1 Arquitectura de Datos a nivel de componentes.

El diseño de datos a nivel de componentes se centra en la representación de estructuras de datos a las que se accede directamente a través de uno o más componentes del software.

Principios para la especificación de los datos:

- Todas las estructuras de datos y las operaciones a llevar a cabo en cada una de ellas deberían estar claramente identificadas.
- Se debe establecer un diccionario de datos y usarlo para definir el diseño de los datos y del programa
- La representación de las estructuras de los datos deberían conocerla solo aquellos módulos que deban hacer uso directo de los datos contenidos dentro de la estructura.

2.2.2 Elementos de la arquitectura de datos

Un problema histórico con el modelado de datos es que todos aquellos responsables de su construcción (principalmente desarrolladores de BD y modeladores de datos) han carecido de un estándar que les ofreciera una guía a seguir en todo el proceso de modelación y desarrollo de la BD, lo que representa un paso de avance, costes de tiempo y de esfuerzo humano cada vez que un nuevo MD es requerido. No existe una plantilla que establezca las pautas a seguir arquitectónicamente a nivel mundial para la vista de datos. De manera que se hace necesario contar con una guía que establezca las bases para el conocimiento teórico y los fundamentos para la puesta en práctica en el modelo de diseño y el desarrollo de las BD.

Partiendo de que no están establecidas las bases para el desarrollo de una AD y que el trabajo en la UCI en relación a este tema no esta explotado, sino que cada proyecto plantea sus propias bases de acuerdo con sus necesidades a la hora de desarrollo, sin generar alguna plantilla o artefacto que lo documente. Para poder establecer un estándar que norme el trabajo de manera general, es necesario contar con una plantilla, que contemple todos los elementos que contempla la AD y que rigen el trabajo con las BD. Entre ellos se destacan:

- Políticas de trabajo.
- Seguridad de los datos.
- Normas de comentariado.
- Estándares de nomenclatura.
    - Nomenclatura de la Base de Datos.
    - Apariencia de los esquemas.
    - Nombres de las tablas.
    - Prefijos a utilizar en la creación de las tablas.

  - Apariencia de los campos.
  - Nombre de las llaves primaria.
  - Nombre de las llaves foráneas.
  - Nombre de las funciones, ***trigger***, tipos de datos y vistas.
  - Nombre de los ***tablespaces***.
  - Nombre de los dominios.
  - Nombre de las secuencias.
- Estándares de tipo de datos.
- Concurrencia.
- Rendimiento.

Cuando la AD es desarrollada y llevada a cabo de una manera correcta, influye notablemente en el desarrollo y calidad del producto que se desarrolle. Con la puesta en práctica de los elementos antes mencionados, se lograría una igualdad a la hora del desarrollo y se haría posible un mejor entendimiento por parte del equipo de trabajo ya que se trabajaría de forma organizada ya que se utilizarían los mismos términos.

2.3 Vista de datos de la arquitectura

En este epígrafe se llevara a cabo la descripción de la vista de datos que se utilizara en la línea CH, de forma tal que se logre unificar el trabajo y mejorar el entendimiento del equipo, ya que se establecen una serie de elementos que serán de uso general. Realizando esto se logra que cada cual trabaje a partir de un estándar.

2.3.1 Políticas de trabajo

- Teniendo en cuenta el uso de réplicas entre bases de datos se debe hacer uso de tipos de datos acordes a dicho requerimiento. No es correcto utilizar el tipo de datos "**int**" para la columna llave primaria de una tabla de usuarios pues al replicar los datos de varias BD en una BD central, entrarían en conflicto los datos (varios usuarios con igual identificador)
- No se debe utilizar la funcionalidad de integración al repositorio subversión presente en el *Visual Paradigm*. No se ha logrado que funcione correctamente ajustándose a las características del proyecto.

- Para hacer cambios en los modelos ubicados en el repositorio, se debe utilizar el modo de trabajo “Bloquea-Cambia-Desbloquea” con los ficheros de los modelos.
- Es importante utilizar el control de versiones para disminuir la posibilidad de pérdida de información y ganar mayor organización en el trabajo.
- Salvar los cambios en el repositorio con la mayor frecuencia posible.
- Dar una descripción breve pero útil de los cambios realizados. Esto ayuda a los desarrolladores a conocer los cambios que se realizaron.
- Siempre actualizar la copia de trabajo desde el repositorio antes de comenzar a realizar cambios importantes en los modelos.
- La integración de las líneas se realizará con la participación de los arquitectos de datos de arquitectura y los otros arquitectos de dato de las líneas correspondientes. Se analizan los cambios que puede afectar a uno o al otro y se efectúa la integración. Esto implica que después que se haga esto los arquitectos de datos no deben cambiar las tablas afectadas porque puede traer conflictos.
- Los índices en la BD serán creados por los arquitectos de datos de arquitectura ya que si se realizan muchos índices en la base de datos puede afectar el rendimiento y el espacio en disco aumentaría considerablemente.
- Para el uso de ***UPDATE*** y ***DELETE*** en ***CASCADA*** se realizará en dependencia de las exigencias del negocio, se debe de tener en cuenta la integridad referencial de las tablas si una depende de otra, o si depende pero no es fuerte la integridad

2.3.2 Seguridad de los datos

- El servicio de BD se brindará mediante el SGBD *PostgreSQL*, mínimo la versión 8.3.
- La base de datos siempre estará protegida con el uso de firewall.
- El control de acceso al servidor se realizará mediante los archivos de configuración de *PostgreSQL* (***postgresql.conf*** y ***pg_hba.conf***).
- Los permisos en la base de datos, esquemas y objetos asociados, se realizarán a través de la creación de roles y grupos de roles, para así darle los permisos de lectura y escritura en dependencia de la responsabilidad o funcionalidad necesaria.

- Se debe utilizar el método de cifrado SSL para una mayor seguridad durante la conexión a la base de datos.
- No se deben utilizar servidores de BD locales para el desarrollo, al usarlo se corre el riesgo de perder la información en caso de dañarse la maquina.
- Cada uno de los subsistemas estará representado con un esquema en la BD, con sus permisos pertinentes.

2.3.3 Normas de comentariado

Es una necesidad comentar todo lo que se haga dentro de la base de datos, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo.

- Los comentarios serán tabulados en forma consecutiva y con el siguiente formato:
  Descripción seguida del 'Nombre de la persona que la realiza'.

  Ejemplo:

  PostgreSQL

  ```
  COMMENT ON FUNCTION "public"."prueba"()
  IS 'comentario de la función ''Pepe''';
  COMMENT ON DOMAIN "public"."domain1"
  IS 'comentario del dominio ''Pepe''';
  ```
- Se deben tratar con sumo cuidado los detalles que no es necesario conocer o que pueden ser deducidos echando un vistazo al código.
- Es importarte ser correcto y no perder el tiempo, comentando solo lo necesario para transmitir la idea. Los comentarios deben ser simples y directos.
- Se comentará mientras se programe, para lograr eliminar mayores costes en el tiempo.
- Los comentarios se actualizarán al mismo tiempo que el código.
- En caso de que se quiera emitir algún comentario especial en cualquier segmento del código utilizar: /* Comentario */.

2.3.4 Estándares de nomenclatura

Apariencia de los esquemas

El nombre del esquema se debe de escribir completo en minúscula, comenzando por el prefijo "**mod**" que significa modulo, seguidamente un guion bajo "_", y finalmente el nombre del modulo.

Ejemplo

'**mod_capitalhumano**';

Nombres de las tablas

El nombre que se empleará en cada tabla debe escribirse con todas las letras en minúscula, para evitar problemas con el *Case Sensitive* del gestor, y consecutivamente sin tener la presencia de guiones u otros caracteres especiales. Además debe ser escrito de forma sencilla pero al mismo tiempo explícita, tal que con solo leerlo se reconozca el propósito de la misma.

Ejemplo

'**dat_persona'**

'**nom_colorojos**'

'**his_hispagosadicionales**'

Como se observa antes del nombre de cada tabla se añade un prefijo, el cual es diferente en dependencia del tipo de tabla que sea. Los prefijos a utilizar son los siguientes:

- **dat_:** Se utiliza en tablas que almacenan la mayor cantidad de datos de la entidad.
- **nom_:** Se utiliza en tablas nomencladoras.
- **his_:** Se utiliza en tablas que almacenan datos por largos períodos de tiempo y que solo son utilizados para análisis esporádicos. Son tablas históricas.
- **seg_:** Prefijo utilizado en tablas que almacenan control de acceso, usuarios y opciones de acceso de uno o varios sistemas. (Tablas de Seguridad)
- **conf_:** Prefijo utilizado en tablas que almacenan parámetros de configuración del sistema. (Tablas de Configuración)

- **tmp_:** Prefijo utilizado para tablas que almacenan datos transitorios. (Tablas Temporales)
- **res_:** Prefijo utilizado para las tablas resúmenes, empleadas en los reportes.

Estructura de los campos

Los nombres se escribirán completamente en minúsculas y al igual que el nombre de las tablas sin presentar guiones ni caracteres especiales. Debe ser explícito de forma tal que con solo leerlo se sepa el propósito del mismo. Cada campo deberá tener una descripción del mismo, de forma que se explique de forma sencilla cual será su función en la tabla, es decir qué es lo que se almacenará. En caso de que sea una llave primaria se le antepondrá el prefijo "**id**" y seguidamente el nombre del campo, aso mismo si es llave foránea.

Ejemplo

'**idpersona'**: Identificador de la persona.

'**idcolorpiel'**: Identificador del color de piel.

'**nombre'**: Nombre de la persona.

'**edad'**: Edad de la persona.

Nombre de las llaves primarias

El nombre de las restricciones se escribe con minúscula. Comienza con el identificador id seguido el nombre de la tabla todo junto y en minúscula.

Ejemplo

idpersona: Llave primaria de la tabla **dat_persona**.

Nombre de las llaves foráneas

El nombre de las llaves foráneas se escribe con minúscula y el nombre de la llave primaria de la tabla donde pertenece.

Ejemplo

idcolorojos: Llave foránea de la tabla "**dat_colorojos**".

Nombres de las funciones, triggers, tipos de datos y vistas

El nombre debe de ser al igual que en el caso de los nombre de las tablas y de los nombre de los campos explícitos, de forma tal que con solos leerlo se reconozca el propósito del mismo. Al igual que en caso de las tablas se utilizaran prefijos según el caso especifico.

'f_': Se utiliza en el caso de las funciones.

'ft_': Se utiliza en el caso de las funciones utilizadas en los ***triggers***.

'**t_':** Se utiliza en el caso de los ***triggers***.

En el caso de los ***triggers*** usados para manejar la concurrencia, los mismos estarán formulados de la siguiente forma.

Empezaran con '**t_**' puesto que son ***triggers***, pero seguidamente le pondrá la palabra '**UPDATE'**, para de esta forma reconocer que el mismo lo que hace es actualizar el campo utilizado para la concurrencia, y posteriormente el nombre del mismo.

'**td_**': Se utiliza en el caso de los tipos de datos.

´**v_**´: Se utiliza para el caso de las vistas.

Nombre de los tablespace

El nombre se definirá todo en minúscula, empezando con la letra **tbs**, seguido un guión bajo y el nombre del módulo junto en minúscula.

Ejemplo

tbs_capitalhumano: Tablespace del módulo de capital humano.

Nombre de los dominios

El nombre se definirá todo en minúscula, empezando con la letra "d", posteriormente el nombre del módulo al que pertenece y por último el nombre del dominio, separados por punto.

Ejemplo

d_modcapitalhumano: Dominio que pertenece al módulo capital humano.

Nombres de las secuencias

Los nombres de las secuencias empezarán con el prefijo '**sec'** seguidamente tendrán un guion bajo '_', posteriormente le continuara el nombre de la tabla a la que perteneces dicha secuencia, después del nombre de la tabla se le añadirá un guion bajo '_', acompañado del sufijo '**seq'**.

Ejemplo

'**sec_persona_seq**'

'**sec_trabajador_seq**'

2.3.5 Estándares de tipos de datos

Los tipos de datos a utilizar esta en dependencia del valor que se almacene en dicho campo, y con el objetivo de lograr un mayor rendimiento de la BD.

Por lo que:

- Los tipos de datos enteros serán representados por el tipo de dato *NUMERIC,* con diferentes niveles de precisión.
  - Para los nomencladores se establece que los identificadores pertenecerán a:

    NUMERIC(1,0) si el crecimiento es 0 < **X** < 10 (**X** $\in$ R)

    NUMERIC(3,0) si el crecimiento es 9 < X < 1000 (X $\in$ R)
  - Para los valores enteros se establecen:

    NUMERIC(5,0) admite enteros de 5 cifras (2 bytes)

    NUMERIC(10,0) admite enteros de 10 cifras (4 bytes)

    NUMERIC(19,0) admite enteros de 19 cifras (8 bytes)
  - Para los valores de coma flotantes, se define:

    NUMERIC con grado de precisión después de la coma de 2 cifras.
  - Para los campos auto incrementables se definirá una secuencia. De manera que no se utilice otro tipo de dato para los valores numéricos.

- Para todos los campos que sean fecha se define:

  DATE representa los tipos de datos fecha.

- Para los valores condicionales (***BOOLEAN***), se define:

  NUMERIC(1,0) para los valores condicionales.

- Para el trabajo con imágenes, se establece:

  BYTEA representado en *Visual Paradigm* por el tipo de dato ***BLOB***.

- Las cadenas serán representadas por el tipo de dato VARCHAR, con diferentes longitudes de cadena.

  VARCHAR(10): Utilizado en campos que no sobrepasen los 10 caracteres. Como por ejemplo el código ONE de las entidades, el color de ojos de una persona, etc.

  VARCHAR(20): Utilizado en campos que no sobrepasen los 20 caracteres.

  VARCHAR(30): Utilizado en campos que no sobrepasen los 30 caracteres. Por ejemplo los nombres de las personas, los cuales no son almacenados en un '***VARCHAR***(20)' porque los mismos pueden ser compuestos.

  VARCHAR(255): Para aquellos campos que almacenan una gran cantidad de caracteres, siempre y cuando no sobrepasen los 255.

  VARCHAR(0): En caso de que nos se sepa la cantidad de caracteres que se necesitará, por ejemplo, un párrafo, un expediente, un libro.

2.3.6 Concurrencia

¿Para que realizar control de Concurrencia?

Un mismo campo puede ser accedido por uno o varios clientes al unísono con el fin de realizar determinada acción; en especial a la hora de realizar una modificación.

Para darle solución a la problemática de que un cliente modifique un dato que ya ha sido modificado sin tener notificación del cambio, se le agregará un campo "version", de tipo ***NUMERIC***(6,0), el cual informará las veces que la tupla se ha modificado.

Mediante un TRIGGER, se comprobará que si la nueva versión que pone el cliente es igual a la antigua versión incrementará en uno y permitirá modificar la tupla accedida, de lo contrario se lanzaría una excepción. Ejemplo del *trigger*.

CREATE TRIGGER "**tr_updatepersona**" ***BEFORE UPDATE***

ON "mod_capitalhumano"."**dat_persona**" ***FOR EACH ROW***

EXECUTE PROCEDURE "public"."chequear"();

2.3.7 Rendimiento

Cuando se diseña e implementa una BD, deben identificarse las tablas de gran tamaño y los procesos más complejos que realizará la base de datos. También se debe prestar una atención especial al rendimiento cuando diseñan estas tablas, además, de considerar los efectos que puede tener en el rendimiento el aumento del número de usuarios con acceso a la BD.

Los siguientes cambios de diseño, pueden mejorar el rendimiento:

- Si una tabla que contiene cientos de miles de filas debe resumirse en un informe diario, puede agregar a la tabla una o varias columnas que contengan datos previamente agregados para utilizarlos sólo en dicho informe.
- Las BD pueden normalizarse en exceso. Esto significa que la BD se define con un gran número de tablas pequeñas interrelacionadas. Cuando la BD procesa los datos de estas tablas, debe realizar muchas más operaciones para combinar los datos relacionados. Este procesamiento adicional puede repercutir negativamente en el rendimiento de la base de datos. En esos casos, una reducción de la normalización de la BD para simplificar procesos complejos puede mejorar el rendimiento.
- Es importante realizar consultas a las tablas de la manera más optima posible, mucho mas cuando en ella interviene clausulas como el ***JOIN***, implicando la unión de varias tablas que se rigen por una condición a cumplir y el ***WHERE***, que determina los campos que entrarían a formar parte de la tabla resultado. En este caso es importante definir cual de estas sentencias se utiliza con prioridad, para reducir las búsquedas y mejorar el rendimiento.

La principal importancia de la vista de datos de la arquitectura, es que brinda en forma de plantilla una serie de pasos comunes a seguir por los arquitectos de datos en pro de lograr uniformidad a la hora de realizar el diseño de los MD.

2.4 Clases persistentes

Las clases persistentes de una aplicación implementan las entidades del problema del negocio. Son aquellas que sus datos perduran en el tiempo y deben almacenarse en la BD. Por el contrario, las clases temporales, son creadas y manejadas por el sistema en tiempo de ejecución, y al terminar este dejan de existir. El diagrama de clases persistentes tiene como entrada, según la metodología *RUP* [3] el diagrama de clases del diseño, a partir del que se definen las clases que van a persistir en la BD.

2.4.1 Descripción de las clases persistentes

Después de haber enunciado que es lo que son las clases persistentes, nos encontramos en condiciones de entrar a especificar las entidades que persistirán en la BD.

Después de un profundo análisis y discusión con los analistas y especialistas funcionales, en conjunto con el equipo de datos, y a partir de los diagramas de clases del diseño se identificaron las clases persistentes. De manera, que se obtuvieron clases candidatas a ser entidades persistentes.

A continuación se mencionarán dichas clases así como su propósito. Para una mayor comprensión y organización, se dividirán las clases agrupándolas por componentes o subsistemas que constituyen la solución, ya que la cantidad de clases es notablemente grande.

Componente Persona

Este componente esta encabezado por la clase **Persona**, que maneja los datos referentes a las personas; como por ejemplo, el número de carnet de identidad, el nombre, los apellidos, el sexo y otras características físicas; además de otras no físicas como, su categoría docente, el grado científico, su procedencia social, el grupo sangre que posee, si es casado o no, su nivel de escolar y grado escolaridad, numero de teléfono si tiene, fecha de nacimiento, si es jimagua o no, además características referentes a la fonética del nombre y los apellidos.

[3] RUP: Proceso Unificado de Desarrollo.

Por otra parte, la Clase **Personatalla**, es una composición la clase Persona y maneja los datos referentes a las tallas, como por ejemplo, las tallas de las prendas de vestir que usan las personas.

Para dar apoyo a la clase **Personatalla** se hace necesario disponer de otras que ayudan a manejar los datos referentes a los distintos tipos de prendas. De manera que se crean las clases: **Gorra**, **Pantalonosaya**, **Calzado** y **Camizaoblusa**.

Para controlar los datos referentes a las características físicas de las personas se crean las clases: **Sexo**, **Colorpelo**, **Colorpiel** y **Colorojos**. Además de otras que apoyan a la clase **Persona**, en las que se relacionan los datos afines a las características no físicas, como son: **Gruposanguineo**, **Categoriadocente**, **Gradocientifico**, **Extracionsocial**, **Estadocivil**, **Gradoescolaridad** y **Nivelescolar**.

Componente Trabajador

Es uno de los componentes más importantes, pues esta relacionado de una forma u otra con los demás componentes. Su clase principal es **Trabajador**. Esta maneja datos como: el número del expediente interno del trabajador, el tipo de contrato, el sistema de pago, el impuesto y la fecha fin si el contrato es determinado, además del movimiento de nómina. El tipo de contrato, el sistema de pago y el movimiento de nómina, son atributos que obtiene de las clases **Contrato**, **Sistemapago** y **Modelomovnomina**, respectivamente.

Modelomovnomina es la clase encargada de manejar los datos referentes a los movimientos realizados a los trabajadores que produce modificaciones en su estructura salarial, cargo o área de trabajo. Entre los datos que manipula se encuentran: la fecha de movimiento, fecha efectiva, fecha fin del movimiento y observaciones del mismo. Además cuenta con los atributos tipo de reubicación y motivo de movimiento de nómina que se obtienen de las clases **Tiporeubicacion** y **Motivomovonomina** respectivamente, este última a su vez tiene atributos que pasan a la clase **Tipomovnomina** producto a la relación que se establece entre estas clases.

Cuenta además con una clase **Datoscontables** que se encarga del manejo de los datos contables del trabajador. Entre los datos que opera están: la fecha en que se realiza, la fecha efectiva en que los datos comienzan a tener efecto, si liquida o no vacaciones acumuladas, y numero de tarjeta si tiene.

Para completar este componente se encuentra la clase **Registropagosadicionalestrabajador**, que es la que manipula los pagos adicionales que puede tener asociado cada trabajador, se encarga de manipular el importe del pago adicional así como la fecha efectiva que tendrá el mismo. Además se cuenta con una clase **Hisregistropagosadicionalestrabajador** que no es más que un historial de los pagos adicionales de a cada trabajador; lo que permite conocer cuales pagos adicionales se le pagaban en el pasado.

Componente Pagos Adicionales

En este componente se procesan todos los datos referentes a los pagos adicionales. Su clase principal, como su nombre lo indica, es **Pagosadicionales** y maneja los datos necesarios para poder gestionar de manera correcta dichos pagos, como es el caso de la denominación y descripción del pago, el código del mismo, su valor, la ecuación, si es para el trabajador o para el puesto de trabajo, la fecha de inicio del pago, y la fecha en la que comienza a ser efectivo.

Cuenta además, con la clase **Categoriapagosadicionales** que gestiona los tipos de pagos adicionales existentes y la clase **Hispagosadicionales** en la cual se tramita un historial de dichos pagos, ya que los pagos adicionales pueden cambiar su importe y es necesario conocer el importe que tenia antes de ser cambiado.

Componente Puesto de Trabajo

Este componente permite definir los puestos de trabajo de la empresa. El puesto de trabajo es el lugar que un trabajador ocupa cuando desempeña una tarea que esta relacionado con el área de trabajo y el cargo. Define los grupos de puestos de trabajo de la empresa. Estos grupos concentran los puestos de trabajos que posean características semejantes. Por otra parte permite definir los pagos adicionales que tienen los grupos de trabajo y por ende los puestos de trabajo que pertenecen a estos grupos.

La clase principal de este componente es **Puestotrabajo**, la cual controla los datos referentes a dicho puesto, como por ejemplo la denominación y el código del mismo, si esta ocupado por algún trabajador o no, el grupo de puesto de trabajo al que pertenece, la fecha de inicio y fin del mismo si la requiere, y el movimiento de nómina que genera.

También cuenta con una clase **Planillapuestotrabajo** que no es más que los grupos de puestos de trabajos similares, la misma manipula, el tipo de cargo, la descripción y el código, el fondo de tiempo del

mismo y la jornada laboral, así como la fecha de creado y eliminado, y la fecha en que comienza a ser efectivo el puesto.

Además posee una clase **Plantillapuestotrabajopagosadicionales** que se encarga del controlar que pagos adicionales le corresponden a cada grupo de puesto de trabajo, además controla el importe de los pagos adicionales y la fecha en que empieza a ser efectivo el pago adicional.

Como parte de este componente también esta la clase **Hisplantillapuestotrabajopagosadicionales**, encargada del manejo de los datos históricos de los pagos adicionales, lo cual es necesario para si ocurriese una modificación en los pagos adicionales que están asociados al puesto de trabajo, conocer cuales eran los valores que tenían los pagos antes de ser modificados.

Componente Incidencia

En este componente se agrupan las clases relacionadas con las incidencias cometidas por los trabajadores. Esta compuesto por cinco clases entre las que se encuentran **Registroincidencia**, que permite manejar todas las incidencias que tiene un trabajador durante un periodo de tiempo. Los datos que manipula son, la fecha de inicio y fin de la incidencia, el importe y las horas de la mismas, además si la incidencia está procesada o no.

Otras de las clases es **Incidencia**, que se encarga de manejar las incidencias como tal, es decir cuando aún no ha sido asociada al trabajador. Entre los datos que opera se encuentran, el código, la denominación, el comportamiento y el porciento de la incidencia, si está solapada o no, si acumula o no vacaciones en días, si acumula o no vacaciones en importe, la fecha de inicio y fin de la incidencia, además del tipo de incidencia a la que pertenece; que es manejado por la clase **Tipoincidencia**, que gestiona las posibles definiciones de los tipos de incidencias que afectan al salario de los trabajadores.

Posee además una clase **Registroimpuesto** que no es más que la asociación de una incidencia con un impuesto. Otra de las clases es **Impuesto** que maneja datos como, el código del impuesto, su denominación, el porciento del mismo, si es de tipo salarial o no, y la fecha de inicio y fin del impuesto.

Componente Nomina

Este componente agrupa todas las clases que ayudan a contabilizar las nóminas para el pago de los trabajadores. Cuenta con una serie de clases para este fin, entre las que se encuentra **Nomina** que es la encargada de manipular los datos de la nómina cuando aún no esta procesada, como es el caso de el

tipo de nómina, el período de pago, el estado de la nómina, la fecha en que se realiza y la descripción de la misa.

Otras clases que se relacionan con **Nomina** son: **Tiponomina**, **Estadonomina**, que manejan los tipo de nómina y los estados de las nóminas respectivamente, además de **Periodopago**, que manipula el períodos de pagos en que se realiza la nómina. Esta clase gestiona los datos del programa que se le aplica al período, la fecha de inicio y fin del período de pago, las horas sueldistas y las horas jornaleros, la denominación y fondo del período.

Otra de las clases es **Docajuste**. Su función es maniobrar los ajustes que se le pueden realizar a la nómina y entre los datos que manipula se encuentran el tipo de ajuste que se le realiza a la nómina, el trabajador a que pertenece la nómina, la nómina a la que se le hace el ajuste, el estado de esta, el importe, la descripción, el tiempo y la cantidad de horas por las cual se realiza el ajuste.

Tipoajustenomina es la clase encargada de manejar los datos de los tipos de ajuste que se pueden realizar y cuenta con atributos como: denominación, código, si acumula o no vacaciones en días y en importe, una cuenta y un elemento de gasto. Además cuenta con la clase **Registroimpuestotipoajuste** que es la encargada de controlar que impuesto se le asocia a un tipo de ajuste.

La clase **Comprobanteoperaciones**, no es más que el comprobante que se le realiza a una nómina. Los datos de dicho comprobante son: la nómina a la que se le realiza, el número del comprobante, la fecha en que se realiza y una descripción del mismo.

La clase **Procesamientonomina** cuya función es manejar los datos referentes al procesamiento de la nómina, es la que más datos maneja dentro del módulo, entre los que aparecen: la nómina que se procesa, el trabajador al que se le realiza la nómina, la fecha de procesamiento de la misma, el salario escala del trabajador, las horas trabajadas, el total de pagos adicionales que se le paga y las retenciones que se le cobra al trabajador, el descuento que se le realiza, el salario bruto, importe y días acumulados en vacaciones, el salario devengado, otros pagos que se le realizan, las contribuciones especiales que posea el trabajador, el fondo, así como quien la elaboró, revisó, contabilizó y aprobó.

Para terminar con las clases de este componente, se mencionan **Procesaminetopagosadicionales**, **Procesamientoincidencia**, **Procesaminetoretenciones**, que manejan los datos referentes al procesamiento de los pagos adicionales, las incidencias y las retenciones de los trabajadores

respectivamente. La primera gestiona el procesamiento de la nómina, los pagos adicionales, el importe, y al trabajador que se le procesan los pagos. La segunda gestiona al igual que la primera el procesamiento de la nómina, el trabajador al cual pertenece la incidencia, el importe, la incidencia que se procesa, las horas y el comportamiento de la misma. Por ultimo y al igual que con las anteriores **Procesaminetoretenciones** registra el procesamiento de la nómina y al trabajador, además del registro de retenciones y el importe de las mismas.

Componente Submayores

En este componente se concentran todas las clases que tienen que ver con los submayores que se generan en la entidad. Entre ellas se encuentra **Submayorretenciones** que gestiona los datos de las diferentes retenciones que se procesan y manipula los datos referentes a la nómina, el registro de retenciones, la operación, la fecha en que se realiza, el debe, el haber y el saldo de la retención, el debe, el haber y el saldo al banco y la referencia de la misma.

Registroretenciones maneja las retenciones específicas del trabajador con las operaciones que han ido afectando dichas retenciones. Entre los datos que tramita están el tipo de retención, el trabajador al que se le aplica, el monto de la misma, el importe por cada plazo, la cantidad de plazos y el importe del ultimo plazo, si esta activa o no, así como la fecha y la referencia de la misma.

Tiporetencion gestiona los datos referentes a los diferentes tipos de retenciones que pueden tener un trabajador como: su denominación, el código y la prioridad de la retención y si la retención tiene o no condición de parada.

Las clases Programa y **Operacionsub** manipulan los distintos programas y operaciones. Asimismo se encuentra **Conceptoretenciones**, que es la encargada de conocer que el registro de retenciones que le pertenece a cada programa, y **Configuracionsubmayorvac** donde se maneja el porciento de vacaciones que se especifique.

La clase **Submayorvac** que gestiona las operaciones que se realizan sobre el submayor de vacaciones y maneja datos como la nómina, el trabajador, la operación, el acumulado anterior en días y en importe, el acumulado del periodo en días y en importe, lo que se le paga al trabajador por vacaciones tanto en días como en importe y la fecha en que se realiza.

Como parte de este componente también están las clases **Hissubmayorvac** y **Hisregistroretenciones** que manejan datos históricos del submayor de vacaciones y del registro de retenciones respectivamente. Estos historiales permiten conocer los datos referentes a las vacaciones y las retenciones que se les pagaron y cobraron respectivamente a los trabajadores en tiempos pasados.

A manera de conclusión se puede expresar, que la clases persistentes son la base para la obtención de un buen MD, mediantes estas se agrupan los atributos de lo que posteriormente serán entidades en la BD, solo tendríamos que conocer que atributos se necesitan que no se encuentran comprendidas en las misma. Para eso es que se hace necesario contar con una matriz de integración que facilite el trabajo de conocer que atributo es el que se necesita y su subsistema de origen.

2.5 Matriz de integración

La matriz de integración es uno de los artefactos que generan los arquitectos de datos, se desarrolla por la presencia en las clases, de datos que son de subsistemas diferentes. Por esta razón se hace necesario contar con algún tipo de documento que sirva de guía al arquitecto de datos a la hora del desarrollo o mantenimiento de la BD.

En el caso específico de la línea de CH, la matriz de integración es de gran ayuda, pues este subsistema necesita información que se encuentra representada en otros subsistemas. Estos datos se utilizan en más de un componente y en más de una clase dentro de cada componente. En esto radica su importancia y aplicabilidad, ya que sirve para tener el control de que subsistema proviene cada dato y donde se utiliza.

Para realizar la matriz se analizaron cada una de las clases persistentes con la que cuenta el subsistema CH. De esta forma se conforma la matriz de integración que se muestra a continuación; donde EC, Nom y Cont son los subsistemas Estructura y Composición, Nomencladores y Contabilidad respectivamente.

Tabla 1: Matriz de Integración

Capital Humano	Otros subsistemas	Tabla	Atributos
Comprobanteoperaciones	EC	nom_filaestruc	idfila

Datoscontables	Nom Cont	dat_centrocomun nom_cuenta	idcentrocomun idcuenta
Docajuste	EC	nom_filaestruc	idfila
Modelomovnomina	EC	nom_filaestruc	idfila
Nomina	EC	nom_filaestruc	idfila
Periodopago	EC	nom_filaestruc	idfila
Procesamientoincidencia	Nom Cont	dat_elementocomun nom_cuenta	idelementocomun idcuenta
Procesamientonomina	Nom Cont	dat_centrocomun nom_cuenta	idcentrocomun idcuenta
Procesaminetopagosadicionales	Nom	dat_elementocomun	idelementocomun
Procesaminetoretenciones	Cont	nom_cuenta	idcuenta
Registroincidencia	Nom Nom	dat_centrocomun dat_elementocomun	idcentrocomun idelementocomun
Registroretenciones	EC	nom_filaestruc	idfila
Submayorvac	EC Cont	nom_filaestruc nom_cuenta	idfila idcuenta
Configuracionsubmayorvac	Cont EC	nom_cuenta nom_filaestruc	idcuenta idfila
Impuesto	EC Cont Nom	nom_filaestruc nom_cuenta dat_elementocomun	idfila idcuenta idelementocomun
Incidencia	Nom Cont EC	dat_elementocomun nom_cuenta nom_filaestruc	idelementocomun idcuenta idfila
Pagosadicionales	Nom EC	dat_elementocomun nom_filaestruc	idelementocomun idfila
Plantillapuestotrabajo	EC	nom_filaestruc	idfila
Programa	EC	nom_filaestruc	idfila
Tipoajustenomina	Nom EC	dat_elementocomun nom_filaestruc	idelementocomun idfila
Tiporetencion	Nom	dat_elementocomun	idelementocomun

La matriz de integración es de gran importancia, pues brinda la posibilidad de agrupar todos los atributos que no son propios del subsistema CH, además permite conocer a que subsistema pertenece dicho atributo.

2.6 Modelo de datos

El objetivo de construir un MD es identificar y representar las tablas (entidades) de importancia para el funcionamiento del negocio, sus propiedades (atributos), y la forma en que estas tablas se comunican entre si (relaciones). Este modelo se desarrolla para facilitar el diseño de la BD, su misión es mostrar los datos que contendrá un sistema como un conjunto de objetos con atributos propios.

2.6.1 Descripción del Modelo de datos

El MD de la línea Capital Humano esta compuesto por 61 tablas, cada una representa una clase persistente, evidenciando el mapeo que se establece entre las partes. Cuenta con 33 tablas nomencladores, que son utilizadas para agrupar datos con características similares ya sean gestionables o no. Otras 23 tablas de datos utilizadas para el almacenamiento de los datos que se gestionan en las empresas y 5 tablas que figuran historiales, donde se almacenan datos históricos de cinco de las tablas de datos, con el objetivo de no tener que estar manipulando todos esos datos en cada consulta sobre las tablas de las cuales guardan su historial.

A continuación se analizan las tablas principales del subsistema CH por su importancia y la función que cumplen, además se describen las relaciones de estas clases. Al igual que se hizo con las clases persistentes, se divide por componentes para lograr una mayor comprensión y organización.

Componente Persona

Este componente posee 17 tablas, entre ellas 2 de datos, 15 tablas nomencladores y 16 relaciones, que se mencionan a continuación. Para ver el MD ver (Anexo #1)

Tablas	Descripción
dat_persona	En esta tabla se guardan los datos de todas las personas que estén relacionadas con el sub-sistema de CH. Datos como: Nombre, Apellidos,

	Sexo, Número de Identidad, entre otros.
dat_personatalla	Se almacenan los datos referentes la talla de la persona. Ejemplo: Talla de camisa, blusa, pantalón, etc.
nom_gradoescolaridad	Almacena los distintos grados escolares existentes existen. Ejemplo: Primer grado, Sexto grado, Decimo grado.

Para ver las demás tablas y sus descripciones ver (Anexo #1)

Para ver las descripciones de los atributos de las tabla ver (Anexo #1)

Relaciones de la Tabla dat_persona.

Llaves foráneas	Descripción
idgruposanguineo	Representa que una persona tiene un grupo sanguíneo.
Idcategoriadocente.	Representa que una persona tiene una categoría docente.
idgradocientifico	Representa que una persona tiene un grado científico.
idextrasocial	Representa que una persona tiene una procedencia social.
idgradoescolaridad	Representa que una persona tiene un grado de escolaridad
idsexo	Representa que una persona tiene un sexo
idcolorpelo	Representa que una persona tiene un color de perlo.
idcolorojos	Representa que una persona tiene un color de ojos.
idcolorpiel	Representa que una persona tiene un color de piel.
idestadocivil	Representa que una persona tiene un estado civil.

Relaciones de la Tabla dat_personatalla.

Llaves foráneas	Descripción
idpersona	Representa que una talla le pertenece a una persona.

idgorra	Representa que talla tiene una talla de gorra.
idcalzado	Representa que una talla tiene un número de calzado.
idcamisaoblusa	Representa que una talla tiene una talla de camisa o blusa.
idpantalonosaya	Representa que una talla tiene una talla de pantalón o saya.

Relaciones de la dat_gradoescolaridad

Llaves foráneas	Descripción
idnivelescolar	Representa a que nivel escolar tiene cada grado de escolaridad.

Componente Trabajador

Este componente posee 10 tablas, 4 de datos, 5 tablas nomencladores, 1 historial y 16 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexo # 2)

Tablas	Descripción
dat_trabajador	Almacena los datos de todos los trabajadores que trabajan en la empresa.
dat_modelomovnomina	Almacena los datos referentes al Modelo de Movimiento de Nómina que se genera cuando se realiza una operación sobre el trabajador que afecte ya sea su salario, su cargo o su puesto de trabajo.
dat_datoscontables	Tabla que almacena todos los datos contables que poseen los trabajadores.

Para ver las demás tablas y sus descripciones ver (Anexo # 2 Tabla # 20)

Para ver las descripciones de los atributos de las tabla ver (Anexo # 2)

Relaciones de la Tabla dat_trabajador.

Llaves foráneas	Descripción
idpersona	Representa que una persona puede ser una o muchas veces trabajador.
idpuestotrabajo	Representa que un trabajador tiene un puesto de trabajo.

idcontrato	Representa que un trabajador tiene un contrato.
impuesto	Representa que un trabajador tiene un impuesto.
idmodelomovnomina	Representa que a un trabajador se le realiza un modelo de movimiento de nómina.
idsistemapago	Representa que a un trabajador se le efectúa un sistema de pago.

Relaciones de la Tabla dat_modelomovnomina.

Llaves foráneas	Descripción
idmotivomovnomina	Representa que modelo de movimiento de nómina tiene un motivo de movimiento de nómina asociado.
idtiporeubicacion	Representa que un modelo de movimiento de nómina tiene un tipo de reubicación.
idestructuracomun	Representa a que entidad pertenece el modelo de movimiento de nómina.

Relaciones de la Tabla dat_datoscontables

Llaves foráneas	Descripción
idtrabajador	Representa que un dato contable es para un trabajador.
idcuenta	Representa que un dato contable le pertenece una cuenta.
idcentrocomun	Representa que a un dato contable le pertenece un centro común.

Componente Pagos Adicionales

Este componente posee 6 tablas, 2 tablas nomencladores, 1 historial y 3 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexo # 3)

Tablas	Descripción

nom_pagosadicionales	Almacena los diferentes pagos adicionales que se le pueden pagar tanto al trabajador como a los puestos de trabajo.
nom_categoriapagosadicionales	Almacena las diferentes categorías que puede tener un pago adicional.
his_hispagosadicionales	Almacena un historial de los pagos adicionales.

Para ver las descripciones de los atributos de cada tabla ver (Anexo # 3)

Relaciones de la Tabla nom_pagosadicionales.

Llaves foráneas	Descripción
idcategoriapagosadicionales	Representa que un pago adicional tiene asociado una categoría de pagos adicionales.
idelementocomun	Representa que un pago adicional tiene asociado elemento de gasto.
idestructuracomun	Representa que un pago adicional pertenece a una entidad.

Componente Puesto de Trabajo

Este componente posee 2 tablas de datos, 1 tablas nomencladores, 1 historial y 9 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexo # 4)

Tablas	Descripción
dat_puestotrabajo	Almacena los diferentes puestos de trabajo existentes en la empresa
dat_plantillapuestotrabajopagosadicionales	Almacena los pagos adicionales asociados al puesto de trabajo.
nom_plantillapuestotrabajo	Se almacenan los datos de los puestos de trabajos similares dentro de la empresa.
his_hisplantillapuestotrabajopagosadicionales	Almacena un historial de los pagos adicionales

	asociados al puesto de trabajo.

Para ver las descripciones de los atributos de cada tabla ver (Anexo # 4)

Relaciones de la Tabla dat_puestotrabajo.

Llaves foráneas	Descripción
idplantillapuestotrabajo	Significa que a un puesto de trabajo se le asocia una plantilla de puesto de trabajo.
idmodelomovnomina	Representa que un puesto de trabajo puede generar o no un modelo de movimiento de nómina.

Relaciones de la Tabla dat_plantillapuestotrabajopagosadicionales

Llaves foráneas	Descripción
idplantillapuestotrabajo	Significa que una plantilla de puesto de trabajos de pagos adicionales puede tener uno o muchos plantillas de puestos.
idpagosadicionales	Representa que a una plantilla de puesto de trabajo de pagos adicionales se le pueden asignar uno o muchos pagos adicionales.
idmodelomovnomina	Significa una plantilla de puesto de trabajo de pagos adicionales puede generar o no un modelo de movimiento de nómina.

Relaciones de la Tabla nom_plantillapuestotrabajo

Llaves foráneas	Descripción
idcargo	Significa que una plantilla de puesto de trabajo tiene un cargo.
idarea	Significa que una plantilla de puesto de trabajo pertenece a un área de trabajo.
idmodelomovnomina	Significa que una plantilla de puesto de trabajo puede generar un modelo de movimiento de nómina.
idestructuracomun	Significa que una plantilla de puesto de trabajo pertenece a una entidad.

Componente Incidencia

Este componente posee 2 tablas de datos, 3 tablas nomencladores y 18 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexo # 5)

Tablas	Descripción
dat_registroincidencia	Se almacenan todas las incidencias que tengan asociadas los trabajadores.
dat_resgistroimpuesto	Se almacenan impuestos que tengan asociados las incidencias.
nom_incidencia	Se almacenan todas las incidencias de la empresa.
nom_impuesto	Se almacenan todos los impuestos de la empresa.
nom_tipoincidencia	Almacena los diferentes tipo que pueden ser las incidencias

Para ver las descripciones de los atributos de cada tabla ver (Anexo # 5)

Relaciones de la Tabla dat_registroincidencia

Llaves foráneas	Descripción
idtrabajador	Significa que un registro de incidencia puede tener uno o muchos trabajadores.
idincidencia	Significa que un registro de incidencia puede tener una o muchas incidencias.
idelementocomun	Significa que un registro de incidencia puede tener ninguno o muchos elementos de gastos.
idcentrocomun	Significa que un registro de incidencia puede tener uno o muchos centros comunes.
idperiodopago	Significa que un registro de incidencia tiene un periodo de pago.
idcuenta	Significa que un registro de incidencia puede tener una o muchas cuentas.

Relaciones de la Tabla dat_registroimpuesto

Llaves foráneas	Descripción
idimpuesto	Significa que un registro de impuesto puede tener uno o muchos impuestos.
idincidencia	Significa que un registro de impuesto puede tener una o varias incidencias.

Relaciones de la Tabla nom_incidencia

Llaves foráneas	Descripción
idtipoincidencia	Representa que una incidencia puede tener uno y solo un tipo de incidencia.
idtiponomina	Representa que una incidencia puede tener un tipo de nómina.
idelementocomun	Significa que una incidencia puede tener ninguno o muchos elementos de gastos.
idcuentacreditovac	Significa que una incidencia puede tener ninguna o muchas cuentas, utilizadas para el crédito de vacaciones.
idelementogasto	Significa que una incidencia puede tener ninguno o muchos elementos de gastos.
idcuentacreditoincidencia	Significa que una incidencia puede tener ninguna o muchas cuentas, utilizadas para el crédito de la incidencia.

Relaciones de la Tabla nom_impuesto

Llaves foráneas	Descripción
idelementocomun	Significa que un impuesto puede tener ninguno o muchos elementos de gastos.
idcuenta	Significa que un impuesto tiene una cuenta.

idotracuenta	Significa que un impuesto tiene ninguna o una cuenta.
idestructuracomun	Significa que un impuesto pertenece a una entidad.

Componente Nomina

Este componente posee 12 tablas, 9 tablas de datos, 3 tablas nomencladores y 33 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexo # 6)

Tablas	Descripción
dat_nomina	Tabla que almacena los datos de Nómina.
dat_periodopago	Almacena los diferentes periodo de pago definidos por la entidad
dat_docajuste	Almacena los Documentos de Ajuste que se le pueden realizar a la nómina.
dat_procesamientonomina	Almacena es el resultado del cálculo de la nómina.

Para ver las demás tablas y sus descripciones ver (Anexo # 6 Tabla # 43)

Para ver las descripciones de los atributos de cada tabla ver (Anexo # 6)

Relaciones de la Tabla dat_nomina

Llaves foráneas	Descripción
idperiodopago	Significa que una nómina se realiza un periodo de pago.
idestado	Significa que una nómina tiene uno estado.
idestructuracomun	Significa que una nómina pertenece a una entidad.

Relaciones de la Tabla dat_periodopago

Llaves foráneas	Descripción
idprograma	Significa que a un periodo de pago se le aplica un programa.

idestructuracomun	Significa que un periodo de pago pertenece a una entidad.

Relaciones de la Tabla dat_docajuste

Llaves foráneas	Descripción
idtipoajuste	Significa que un Documento de ajuste se le realiza un tipo de ajuste.
idtrabajador	Representa al Trabajado al que pertenece la nómina a la que se le realiza el Documento de ajuste.
idnomina	Significa que a una Documento de ajuste se le hace a una nómina.
idestado	Represente el estado que presenta en Documento de ajuste.
idnominilla	Representa que nómina esta procesada
idestructuracomun	Significa que el Documento de ajuste pertenece a una entidad.

Relaciones de la Tabla dat_procesamientonomina

Llaves foráneas	Descripción
idnomina	Significa que nómina es la que esta procesada.
idcentrocosto	Representa a que centro de costo pertenece la nómina que se procesa.
idcuenta	Representa la cuenta que tiene asociada la nómina que se procesa.

Componente Submayores

Este componente posee 4 tablas de datos, 4 tablas nomencladores, 2 tablas historicas y 17 relaciones, las cuales se mencionan a continuación. Para ver el MD ir a (Anexos # 7)

Tablas	Descripción
dat_submayorretenciones	Se almacenan los datos de lo que se le descuenta al trabajador por deudas contraídas con el banco, estas deudas pueden ser: Ley

	general de la vivienda, Pensión alimenticia, entre otras.
dat_submayorvac	Almacena el acumulado en día y en importe que cada trabajador va acumulando para el disfrute de sus vacaciones.
dat_registroretenciones	Almacenas las retenciones asociadas a cada trabajador.
nom_programa	Almacena los periodos de pagos que están definidos en cada empresa, estos pueden ser semanal, quincenal o mensual

Para ver las demás tablas y sus descripciones ver (Anexo # 7 Tabla # 56)

Para ver las descripciones de los atributos de cada tabla ver (Anexo # 7)

Relaciones de la Tabla dat_submayorretenciones

Llaves foráneas	Descripción
idnomina	Representa a la nómina que actualiza el submayor de retenciones.
idoperacion	Representa la operación que se realiza sobre el submayor de retenciones.
idregirtroretenciones	Representa las retenciones que tiene asociadas el submayor de retenciones.

Relaciones de la Tabla dat_submayorvac

Llaves foráneas	Descripción
idnomina	Representa a la nómina que actualiza el submayor de vacaciones.
idtrabajador	Significa que Trabajador pertenece al submayor de vacaciones.
idcuenta	Representa a la cuenta que se asocia al submayor de vacaciones.
idoperacion	Significa la operación que se realiza sobre el submayor.
idestructuracomun	Representa a que entidad pertenece el submayor de vacaciones.

Relaciones de la Tabla dat_registroretenciones

Llaves foráneas	Descripción
idtiporetencion	Significa que los tipos de retenciones que puede tener el registro de retenciones.
idtrabajador	Representa al trabajado que posee retenciones.
idestructuracomun	Significa que el registro de retenciones pertenece a una entidad.

Relaciones de la Tabla nom_programa

Llaves foráneas	Descripción
idestructuracomun	Significa que un programa esta definido en una entidad.

Conclusiones

En este capitulo se describió brevemente que es la AS y se mencionaron los tipos más utilizados de esta por su importancia en la realización de proyectos de grandes dimensiones, especificándose en una de las ramas de estas, la AD, la cual al ser llevada a cabo de manera eficiente hace posible que se obtenga un producto con una elevada calidad. Posteriormente y después de analizadas las clases del diseño se presentaron las clases persistentes, identificándose que una serie de atributos que están manipulan no pertenecen a al subsistema CH. Por ello se hizo necesario contar de una matriz de integración que ayude de una forma u otra a organizar el trabajo, y que garantice saber en todo momento de que subsistema se necesita algo y que es lo que se necesita. Consumiendo este capitulo emerge el MD, del cual se describen sus entidades y sus atributos.

CAPITULO 3: Análisis y discusión de los resultados

Introducción

En el capítulo anterior se obtuvo una solución para el diseño de la BD del subsistema CH del sistema Cedrux. En el presente capítulo se comienza con una explicación detallada de diferentes conceptos a tener en cuenta en el trabajo con las BD, una vez esta son creadas, como la seguridad de la información en el diseño propuesto, profundizando en todos los temas relacionados con la integridad de los datos; además de la normalización, dejando claro el nivel hasta donde se recomienda normalizar y en el que se encuentra el MD realizado, y finalmente el tema de la redundancia de los datos como puede afectar el rendimiento de una BD y en especial la que ha sido modelada. Por otra parte, se realizan pruebas de concepto para realizar el análisis de los tiempos de respuesta, discutir los resultados obtenidos y poder arribar a conclusiones. De manera general, se describe el tratamiento dado a aspectos de vital importancia para el área de la BD como son la integridad, la normalización, redundancia y seguridad de la información en el diseño propuesto, elementos donde se han centrados los esfuerzos.

3.1 Integridad de datos

Se refiere a la corrección y completitud de los datos en una BD. Cuando el contenido de una BD se modifica con sentencias *INSERT*, *DELETE* o *UPDATE*, la integridad de los datos almacenados puede perderse de las siguientes maneras:

- Puede añadirse información no válida a la BD; por ejemplo, la talla de una persona asociada a una persona que no existente.
- Pueden modificarse datos existentes tomando un valor incorrecto; por ejemplo, si se asigna a una persona a un color de ojos que no existe.
- Pueden crearse entidades en la base de datos con igual identificador repetidos en la misma tabla; ejemplo, si existe un número de expediente de un trabajador igual a otro, cuando este tiene que ser obligatoriamente diferente.

De manera general se pueden citar varios ejemplos que demuestran la necesidad de establecer y definir la integridad en el momento que se diseñan las bases de datos, ya que esta garantiza ineludiblemente la calidad y consistencia de los datos almacenados.

Según algunas bibliografías y el criterio de algunos especialistas, los temas de integridad se pueden encontrar con los siguientes nombres, integridad de entidad, integridad de dominios, integridad referencial, aunque es importante señalar que pueden ser encontrados en la literatura con nombres diferentes.

3.1.1 Integridad de entidad

La integridad de entidad define una fila como entidad única para una tabla determinada. Ningún atributo de una clave primaria toma valores nulos y a su vez estos valores son distintos para cada posible tupla. Por ejemplo en la solución que se da, si se especifica para una persona que el valor del identificador es **123**, la BD no debe permitir que ninguna otra persona tenga el mismo valor, o sea que no exista otro identificador con ese número.

En el caso del subsistema CH se llevó a cabo la IE de la siguiente manera:

- Se definió que cada una de las tuplas tuviese una llave primaria y que no admitiese valores nulos. Por ejemplo muchas de las llaves primarias son generadas mediante secuencias y van en incremento desde un valor inicial fijado, asegurando que cada llave primaria sea única.
- En las entidades que las llaves primarias no son generadas mediantes secuencias se definió que las llaves fuesen únicas mediantes restricciones *UNIQUE*, o restricciones *PRIMARY KEY*.

3.1.2 Integridad de dominios

La integridad de dominio es la validación de las entradas en una determinada columna. Se puede asegurar la integridad de dominio restringiendo el tipo (a través de tipos de datos), el formato (a través de las restricciones **CHECK** y de las reglas), o el rango de valores posibles (a través de restricciones ***FOREIGN KEY***, restricciones ***CHECK***, definiciones ***DEFAULT***, definiciones ***NOT NULL***, y reglas)

Por ejemplo la columna nombre de la tabla **dat_persona** a la que se pronostica valores entre 1 y 30 caracteres, no se acepta en dicho campo valores con más cantidad de caracteres. Por otro lado la tabla **dat_datoscontables**, que posee una columna de tipo de dato numérico en la que se almacene el número de tarjeta del trabajador, dicho número de tarjeta es un número de 16 caracteres y no se acepta en dicho campo un número de mayor longitud. De igual forma ocurre con los campos que no pueden contener valores nulos y se garantiza esto mediante la restricción ***NOT NULL***.

3.1.3 Integridad referencial

La integridad referencial contempla las relaciones definidas entre tablas, cuando se insertan, modifican o eliminan registros. Está basada en interrelaciones entre claves ajenas y claves primarias o entre claves ajenas y claves únicas (a través de la restricción ***FOREIGN KEY*** y ***CHECK***). La integridad referencial asegura que los valores de las claves son consistentes a través de distintas tablas. Tal consistencia requiere que no exista referencia a valores inexistentes y que, si un valor clave cambia, todas las referencias cambien consistentemente a lo largo de la BD, lo que se conoce como la clausula ***CASCADE.***

La aplicación de este concepto esta presente en el MD propuesto y se refleja en el siguiente ejemplo: En la tabla **dat_persona** existen las llaves foráneas idcolorojos e idcolorpiel las cuales son llaves primarias en **nom_colorojos** y **nom_colorpiel respectivamente**, y mediante la restricción ***FOREIGN KEY*** se evita que se pueda:

- Agregar registros de color de ojos y color de piel a la tabla **dat_persona** si con anterioridad no existen dichos registros en las sus tablas correspondientes.
- No se pueden borrar los registros de color de ojos y color de piel de sus tablas si existen registros relacionados en la tabla **dat_persona**.

Con en el manejo de la integridad de los datos en todas sus presentaciones se logró alcanzar una BD con un control total sobre sus datos, lográndose que no hubiese pérdida de información y evitando que se pudiesen borrar registros de sus tablas propietarias que ya estuviesen registrados en otras tablas. Se evito la presencia de valores nulos en las llaves primarias, además que estas fuesen únicas, proporcionando de esta manera que cada registro tuviese un identificador diferente. Además se tuvo un control estricto sobre los tipos de datos y la longitud de los campos.

3.2 Normalización de la base de datos

La normalización se utiliza para crear relaciones lógicas apropiadas entre las tablas de la BD. Ayuda a prevenir errores lógicos en la manipulación de datos, facilita agregar nuevas columnas sin romper el esquema actual ni las relaciones.

Su intención es reducir las inconsistencias y redundancias de los datos, facilitar el mantenimiento y evitar las anomalías en las manipulaciones de los mismos. El objetivo es obtener un modelo lógico normalizado que represente las entidades normalizadas y las interrelaciones existentes entre ellas.

A mayor nivel de normalización, mayor calidad en la organización de los datos y menor peligro para la integridad de los datos. Este procedimiento consiste en ir alcanzando formas normales. Normalmente, es recomendable alcanzar la máxima Forma Normal (FN), aunque luego es muy probable que restricciones existentes de algún tipo, obliguen a retroceder a un nivel inferior de normalización, o incluso a cierto nivel de des-normalización.

A continuación se mencionan las formas normales básicas que casi todas las bibliografías y autores describen de suficientes a la hora de la normalización:

- **1FN:** si no contiene grupos repetitivos, es decir, todos los atributos dependen funcionalmente de la clave.
- **2FN:** si está en 1FN y cada atributo que no pertenezca a la clave tiene una dependencia funcional completa de la clave.
- **3FN:** si está en segunda forma normal y cada atributo que no pertenezca a la clave no depende transitivamente de dicha clave, en otras palabras, si cada uno de los atributos de la entidad dependen sólo de la clave.

Al modelarse la propuesta de la BD del subsistema CH, se ha deseado evitar puntos que creen confusión y duplicación de la información, así como un mal funcionamiento de la BD. Las anomalías que se trataron de evitar mediante la normalización fueron:

- Redundancia o repetición de los datos en el sistema.
  Para evitar esto se agruparon los posibles datos que se pudiesen repetir en cada una de las tablas y se agruparon en tablas separadas. Ejemplo de esto es el caso de la tabla **dat_persona**, en la cual se repetirían en varias de las tuplas el color de los ojos, el color de piel y el color de pelo, por ello se decidió agrupar estas características en tablas separadas.
- Anomalías de inserción o imposibilidad de adicionar datos en la BD debido a la ausencia de otros datos.
  Para evitar esto y siguiendo la misma secuencia del ejemplo que se describió anteriormente, después de haberse separados los datos específicos de las personas en **dat_persona** y las de

los colores de ojo, pelo y piel, en las tablas **nom_colorojos**, **nom_colorpelo** y **nom_colorpie**l respectivamente, se agregaron en la tabla **dat_persona** como llaves foráneas, columnas correspondiente a cada una de las llaves primaria de las tablas **nom_colorojos**, **nom_colorpelo** y **nom_colorpiel**, evitándose de esta manera que se puedan agregar registros en la tabla **dat_persona** que dependan de que ya existan en las demás tablas.

- Anomalías de borrado o perdidas no intencionadas de datos.
  Se evita al igual que en el ejemplo anterior con la presencia en las tablas de llaves foráneas, por ejemplo, en la tabla **dat_trabajador** se define como llame foránea una columna **idpersona** que a su vez es llave primaria en **dat_persona**, y se evita así que se pueda eliminar una persona que ya haya sido referenciada en la tabla **dat_trabajador**.

Como resultado de la normalización realizada se obtuvo una propuesta de diseño lo más libre posible de anomalías en la actualización y mejora en la independencia de los datos. No se fue tan estricto durante el transcurso de normalizar la BD, con el objetivo de lograr alcanzar un mayor rendimiento a la hora de la realización de las consultas en el menor tiempo posible, por lo que fue necesario en algunos casos duplicar datos, perdiendo en espacio pero ganando en rapidez.

3.3 Análisis de la seguridad de la BD

La información que almacena una BD, está en constante riesgo de sufrir ataques que puedan provocar su modificación o pérdida. Por ello es de vital importancia velar por la seguridad de la BD, protegiéndola contra accesos no autorizados o cualquier acción que puedan violar la integridad de los datos o la confidencialidad de los mismos.

Es de vital importancia garantizar una buena seguridad de los datos, para ello se crearon diferentes roles cada uno con diferentes privilegios. Además se tomaron una serie de medidas para lograr la seguridad requerida, entre las medidas tomadas están:

- Los usuarios no deben compartir sus cuentas ni sus contraseñas.
- Habilitar sólo los servicios y puertos requeridos.
- La cuenta de root y de administrador sólo la debe tener el personal requerido.
- El número mínimo de caracteres de las contraseñas de los usuarios será de 7 caracteres.
- La contraseña del usuario no debe ser igual al nombre del usuario.

- La contraseña debe tener fortaleza requerida con la utilización de letras, números y caracteres especiales.
- Activar la encriptación de contraseñas de la base de datos a la hora de conectarse en el archivo **postgresql.conf** cambiando la propiedad **password_encryption** de la siguiente forma: **password_encryption = on**.
- Crear un usuario de base de datos por cada subsistema.

En el caso de la BD propuesta la seguridad se garantiza utilizando el archivo **pg_hba.conf**, en el mismo se define que PCs (dirección o direcciones IP) tendrán acceso, además a cual o a cuales BD y el modo en que podrán conectarse, que puede ser: conexión con contraseña, validando el usuario y la contraseña para conectarse, o que rechace cualquier conexión de el IP o los rangos de IP configurados y usuarios seleccionados. Además utilizando las opciones del archivo **pg_ident.conf**, permite configurar a que BD pueden acceder determinados usuarios.

3.4 Optimización. Índices y Tuning

Con el objetivo comprobar que la BD cumple con el propósito por el que se desarrolló y validar de alguna manera el trabajo realizado, es preciso realizarle algunos procedimientos de pruebas. Para ello fue necesario adicionarle datos a la BD, sin violar la integridad de la misma y realizarle consultas con cierto grado de complejidad.

3.4.1 Pruebas de concepto

Para llenado de los datos se utilizó la herramienta *EMS Data Generator 2005 for PostgreSQL*, la cual constituye una herramienta poderosa en la generación de datos de pruebas para BD construidas en *PostgreSQL*. Permite seleccionar las tablas para la generación de los datos, definir para cada campo el rango de valores admisibles, además permite llenar la BD de forma eficiente y rápida teniendo en cuanta la integridad referencial de los datos.

Se generaron para la realización de las pruebas de concepto, un volumen de 5000 tuplas para cada una de las tablas de datos y para algunos de los nomencladores más significativo como son los casos de **nom_incidencia**, **nom_plantillapuestotrabajo, nom_tipoajustenomina**, **nom_pagosadicionales**, **nom_tiporetencion**, **nom_programa**, **nom_impuesto**. Estos 5000 tuplas representan una muestra

pequeña, pero significativa de lo que seria la población real de la BD. Los demás nomencladores fueron llenados con datos reales, promediando estos entre 5 y 20 tuplas aproximadamente.

Es necesario tener presente que los resultados obtenidos al realizar las pruebas solo muestran un estimado de la realidad, ya que existen numerosos factores que influyen en la rapidez y estabilidad de la BD, como por ejemplo: la cantidad de usuarios conectados simultáneamente y el grado de concurrencia de las solicitudes realizadas por los mismos. Por otra parte, las pruebas no fueron realizadas sobre un servidor de BD profesional, sino sobre un PC que actuaba como tal, pero de cualquier forma la solución fue liberada para despliegue por el equipo principal de arquitectura central. Las propiedades físicas de la PC utilizada como servidor de BD eran las siguientes: Pentium (R) 4, CPU a 3.00 GHz, 512 MB de RAM, Sistema Operativo Microsoft Windows XP Service Pack 2. El gestor utilizado para realizar la consulta fue *SQL Manager Lite for PostgreSQL*.

Las consultas generadas para la realización de las pruebas, fueron seleccionadas de acuerdo a las operaciones que se esperan sean las más utilizadas una vez implantada la BD y las que más complejidad tienen debido a la cantidad significativa de tablas que se consultan. Además se analizaron las consultas de manera general que se hacían sobre tablas muy grandes que crecerían mucho en el tiempo o aquellas que son ejecutadas con mayor frecuencia.

Consultas realizadas Consulta # 1: Mostrar todas las retenciones pagadas de una entidad en un periodo de pago determinado, dado el periodo de pago y la entidad.

```
SELECT
drr.idestructuracomun,  dpp.denom, est.denominacion as Entidad,
est.codigo as codigoentidad, dt.numexpediente,
dp.nombre, dp.papel, dp.sapel, ntr.denom as tiporetencion,
np.denom as programa, dpp.denom as peridopa, dpr.importe

FROM mod_capitalhumano.dat_procesamientoretenciones dpr
inner join mod_capitalhumano.dat_trabajador dt ON dpr.idtrabajador = dt.idtrabajador
inner join mod_capitalhumano.dat_persona dp ON dt.idpersona = dp.idpersona
inner join mod_capitalhumano.dat_registroretenciones drr ON dpr.idregistroretenciones
                                                          = drr.idregistroretenciones
inner join mod_capitalhumano.nom_tiporetencion ntr ON drr.idtiporetencion = ntr.idtiporetencion
inner join mod_capitalhumano.dat_conceptoretenciones dcr ON drr.idregistroretenciones
                                                          = dcr.idregistroretenciones
inner join mod_capitalhumano.nom_programa np ON dcr.idprograma = np.idprograma
inner join mod_capitalhumano.dat_periodopago dpp ON np.idprograma = dpp.idprograma
INNER JOIN mod_estructuracomp.dat_estructura est on drr.idestructuracomun = est.idestructura
where drr.idestructuracomun = '100000076' AND dpp.denom = 'Marzo primera quincena'
```

Consulta # 2: Mostrar un listado de todos los trabajadores de una entidad, dada la entidad

```
select
est.denominacion as Entidad, est.codigo as CodigoEntidad, op.denominacion as AreaTrabajo,
dp.numid as Expediente, dp.nombre as Nombre, dp.papel as PrimerAP, dp.sapel as SegundoAP,
ncc.dencargociv as Cargo, nco.dencategocup as CatOcupac, ngc.denominacion as GrupoComplejidad,
ns.salario as salarioescala, npa.denom as PagosAdicionales, dpn.salariodevengado as Salario,
nc.denom as TipoContrato

from mod capitalhumano.nom plantillapuestotrabajo nppt
inner join mod capitalhumano.dat puestotrabajo dpt on nppt.idplantillapuestotrabajo = dpt.idplantillapuestotrabajo
inner join mod capitalhumano.dat trabajador dt on dpt.idpuestotrabajo = dt.idpuestotrabajo
inner join mod capitalhumano.dat persona dp on dt.idpersona = dp.idpersona
inner join mod capitalhumano.dat plantillapuestotrabajopagosadicionales dppt on nppt.idplantillapuestotrabajo
                                                                              = dppt.idplantillapuestotrabajo
inner join mod capitalhumano.nom pagosadicionales npa on dppt.idpagosadicionales = npa.idpagosadicionales
inner join mod capitalhumano.dat procesamientonomina dpn on dt.idtrabajador = dpn.idtrabajador
inner join mod capitalhumano.nom contrato nc on dt.idcontrato = nc.idcontrato
inner join mod estructuracomp.dat estructura est on nppt.idestructuracomun = est.idestructura
inner join mod estructuracomp.dat estructuraop op on est.idestructura = op.idestructura
inner join mod estructuracomp.dat cargo dc ON nppt.idcargo = dc.idcargo
inner join mod estructuracomp.dat cargocivil dcc ON dc.idcargo = dcc.idcargo
inner join mod estructuracomp.nom cargocivil ncc ON dcc.idcargociv = ncc.idcargociv
inner join mod estructuracomp.nom categocup nco on ncc.idcategocup = nco.idcategocup
inner join mod estructuracomp.nom grupocomple ngc on dcc.idgrupocomple = ngc.idgrupocomplejidad
inner join mod estructuracomp.nom salario ns on dcc.idsalario = ns.idsalario
where nppt.idestructuracomun = '100000042'
```

3.4.2 Análisis de las pruebas de concepto

Con las pruebas de conceptos realizadas anteriormente, se trata de comprobar que el diseño de la BD cumple con tiempos de respuesta bajos, que constituye la base para garantizar que la BD funcione eficientemente. Se realizaron dos consultas con un elevado nivel de complejidad debido al gran número de tablas que se debían consultar en cada consulta, además, varias de las tablas consultadas no pertenecen al esquema de CH, sino a otros subsistemas que se interrelacionan con este e integran en el sistema Cedrux.

Previamente a la realización de las consultas de prueba se habían eliminado los índices de las tablas que intervendrían en ambas consultas. La eliminación de los índices se llevó a cabo con el objetivo de analizar qué tan eficiente eran estos y se obtuvieron los siguientes resultados.

Consulta	Índices en las tablas	Tuplas devueltas	Tiempo de respuesta aproximado
Consulta 1	No	33	45 ms
Consulta 2	No	57	55 ms

Posteriormente fueron añadidos nuevamente los índices correspondientes a cada una de las tablas y los resultados obtenidos fueron los siguientes:

Consulta	Índices en las tablas	Tuplas devueltas	Tiempo de respuesta aproximado
Consulta 1	Si	33	16 ms
Consulta 2	Si	57	40 ms

Como se puede apreciar en las tablas anteriores, al comparar los resultados de los tiempos de respuesta de la ejecución de las consultas, se llegó a conclusión que el tiempo de respuesta de la BD es significativamente bajo. Además se pudo notar también que cuando fueron agregados los índices mejoró notablemente el rendimiento de la misma.

3.4.3 Normas básicas de optimización.

Como se ha podido comprobar a lo largo de toda la investigación una de las tareas más importantes de las propias de un desarrollador de BD es la de optimización, ajuste o tunning, o sea el mantenimiento de la BD logrando tenerla a punto. Hay que tener en cuenta que las sentencias SQL pueden llegar a ser muy complejas y conforme el modelo de BD vaya creciendo, estas se convierten en complejas y confusas. Es difícil escribir la sentencia correcta la primera vez que se hace, por lo que después de tener cada uno de los procesos modelados, hay que pasar por una etapa de tunning en la que se revisan

todas las sentencias SQL para poder optimizarlas conforme a la experiencia adquirida por cada desarrollador o por ciertas normas que se establezcan.

Después de realizar las pruebas de concepto en la BD, en relación con las consultas y funciones, se pueden establecer algunos elementos que se deben tener en cuenta para garantizar cierto grado de seguridad, menores tiempos de respuesta, menor consumo a nivel de gestor, en fin, lograr un rendimiento acorde con las prestaciones que tendrá la solución que se implementará. Por tanto, ya sea por la cantidad como por la complejidad, la mayoría de las optimizaciones se deben hacer sobre las sentencias *SELECT*, ya que son (por regla general) las responsables de la mayor perdida de tiempo.

A continuación se dan algunas normas básicas a las que se pudo arribar a partir de la experiencia del investigador, con la realización de este trabajo para escribir sentencias *SELECT* optimizadas.

1 Hay que optimizar dos tipos de instrucciones: las que consumen mucho tiempo en ejecutarse, o aquellas que no consumen mucho tiempo, pero que son ejecutadas muchas veces.
2 Si la aplicación funcionaba rápido y se vuelve lenta, se debe parar y analizar los factores que han podido cambiar y hacer que esto ocurra. Si el rendimiento se degrada con el tiempo, es posible que sea un problema de volumen de datos y sean necesarios nuevos índices para acelerar las búsquedas. Pero de cualquier forma es importante tener presente que el añadir un índice equivocado puede ralentizar ciertas búsquedas. Cuantos más índices tenga una tabla, más se tardará en realizar inserciones y actualizaciones sobre la tabla, aunque más rápidas serán las consultas. Por lo que se hace necesario, buscar un equilibrio entre el número de índices y su efectividad, de tal modo que se cree el menor número posible, pero sean utilizados el mayor número de veces posible.
3 Utilizar siempre que sea posible las mismas consultas. La segunda vez que se ejecuta una consulta, se ahorra mucho tiempo de *parsing* y optimización, así que se debe intentar utilizar las mismas consultas repetidas veces.
4 Las condiciones (tanto de filtro como de ***JOIN***) deben ir siempre en el orden en que esté definido el índice. Si no hubiese índice por las columnas utilizadas, se puede estudiar la posibilidad de añadirlo, ya que tener índices extra sólo penaliza los tiempos de inserción, actualización y borrado, pero no de consulta.

5 Para chequeos, siempre es mejor crear restricciones (***CONSTRAINTS***) que disparadores (***TRIGGERS***).

6 Los filtros de las consultas deben ser lo más específicos y concretos posibles. Es decir: es mucho más específico poner ***WHERE*** campo = 'a' que ***WHERE*** campo ***LIKE*** '%a%'. Es muy recomendable utilizar siempre consultas que filtren por la clave primaria u otros campos indexados.

7 Evitar la condiciones ***IN*** (***SELECT***…) sustituyéndolas por ***JOINS***. Cuando se utiliza un conjunto de valores en la clausula ***IN***, se traduce por una condición compuesta con el operador ***OR***. Esto es lento, ya que por cada fila debe comprobar cada una de las condiciones simples. Suele ser mucho más rápido mantener una tabla con los valores que están dentro del ***IN*** y hacer un ***JOIN*** normal. Por ejemplo, esta consulta:

SELECT * FROM dat_persona WHERE nombre IN ('Alberto', 'Alex',... , 'Pedro', 'Juan');

Se puede sustituir por la siguiente consulta, siempre que la tabla **dat_persona** contenga una fila por cada valor contenido en el conjunto del IN:

SELECT * FROM dat_persona dp, dat_trabajador dt WHERE dp.idpersona = dt.idpersona;

También se recomienda tener cuidado cuando se pone un ***SELECT*** dentro del ***IN***, ya que esa consulta puede retornar muchas filas y se estaría cayendo en el mismo error. Normalmente, una condición del tipo "***WHERE*** campo ***IN*** (***SELECT***...)" se puede sustituir por una consulta con ***JOIN***.

8 Cuando se hace una consulta a varias tablas (multi-tabla) con ***JOINS***, el orden en que se ponen las tablas en el **FROM** influye en el plan de ejecución. Aquellas tablas que retornan más filas deben ir en las primeras posiciones, mientras que las tablas con pocas filas deben situarse al final de la lista de tablas.

9 Si en la cláusula ***WHERE*** se utilizan campos indexados como argumentos de funciones, el índice quedará desactivado. Es decir, si tenemos un índice por un campos IMPORTE, y utilizamos una condición como ***WHERE ROUND***(IMPORTE) > 0, entonces el índice quedará desactivado y no se utilizará para la consulta.

10 Una condición negada con el operador ***NOT*** desactiva los índices.

11 Para comprobar si existen registros para cierta condición, no se debe hacer un ***SELECT COUNT***(*) FROM X WHERE xxx; sino que se hace un ***SELECT DISTINCT*** 1 ***FROM*** X ***WHERE*** xxx. De este modo se evita al servidor que cuente los registros.

La base sobre la que se puede lograr realizar optimizaciones a las bases de datos y altos niveles de rendimiento, es la existencia de un buen diseño. Para esto es necesario que la base de datos esté normalizada. Por lo que ciertamente la mejor optimización que se puede realizar en este punto es rediseñar y normalizar la base de datos. Las bases de datos relacionales están diseñadas para funcionar lo más rápidamente posible para un buen diseño relacional, pero con diseños erróneos, se vuelven muy lentas. La mayoría de los problemas de rendimiento tienen un problema de fondo de mal diseño y muchos de ellos no podrán ser optimizados si no se rediseña el esquema de base de datos.

Conclusiones

En este capitulo se realizó un análisis de la solución propuesta. Se analizaron aspectos como la integridad de datos, y dentro de esta, la integridad de entidad, de dominio y la integridad referencial, describiéndose los pasos que se llevaron a cabo para cumplir con cada una de ellas. Se expuso muy brevemente en que consiste el proceso de normalización de la BD; además se expusieron las anomalías que se lograron evitar mediante la normalización y lo que se realizó para alcanzarlo. Se tocaron además temas relacionados con la seguridad de la BD y con la optimización de la misma. Se realizaron diferentes pruebas de conceptos con el objetivo de comprobar el rendimiento de la BD; mediante las pruebas se evidenció que los tiempos de respuestas son bajos y están acorde a las características de la PC usada como servidor.

CONCLUSIONES GENERALES

- Se estudiaron los diferentes MD que se pueden realizar mediante el diseño de BD, haciendo especial énfasis en el estudio del modelo entidad-relación, a partir del que se implemento la solución propuesta.
- Se estudiaron diferentes herramientas *CASE* utilizadas en el diseño de las BD, puntualizando en los aspectos más significativos del *Visual Paradigm*, por ser la herramienta seleccionada por el proyecto para la realización del diseño y modelación de la BD.
- Se estudiaron los diferentes SGBD existentes, clasificándolos de acuerdo a su accesibilidad, en libres, gratuitos y propietarios. Se describieron las características de los SGBD libres y especialmente se detalló el *PostgreSQL*, por ser el gestor de BD seleccionado por el proyecto.
- Se estudiaron diferentes temas referentes al rendimiento y la optimización de las BD, centrando el estudio fundamentalmente en los índices. Llegándose a la conclusión que estos mejoran significativamente el rendimiento pero que no es aconsejable abusar de ellos.
- Se estudiaron temas generales relacionados con la AS y se especificó principalmente en la AD por ser esta la base del trabajo realizado.
- Se plantearon una serie de aspectos importante a tener en cuenta durante el proceso de diseño de la BD, como es el caso de la política de trabajo, normas de comentariado, estándares de nomenclatura, concurrencia y rendimiento.
- Se analizaron cada una de las clases candidatas a convertirse en entidades persistentes, realizándose posteriormente la realización y descripción del MD, quedando comprendido por un total de 61 tablas, 23 tablas de datos, 33 nomencladoras y 5 historiales
- Se analizó el resultado de la solución propuesta, comprobándose que el diseño realizado cumple con los requisitos de multi-entidad y concurrencia; además se realizaron pruebas de concepto, las cuales devolvieron tiempos de respuestas relativamente bajos.
- Se realizaron pruebas de conceptos con el objetivo de comprobar la calidad del diseño de la BD realizada, arrojando estas resultados positivos en dependencia a los tiempos de respuesta obtenidos.

RECOMENDACIONES

- Utilizar este trabajo como guía de estudio para futuros trabajos de diseño de bases de datos dentro del proyecto ERP-Cuba, y que sirva de base para el desarrollo de próximas versiones del subsistema Capital Humano.
- Para una segunda fase se recomienda profundizar en los temas de normalización, con el objetivo de perfeccionar el modelo de datos. para dar respuesta al tema de multi entidad.
- Estudio de temas relacionados con clúster de bases de datos, con el fin de aplicar estos en el subsistema Capital Humano y el sistema Cedrux en general, con el objetivo de lograr mayor seguridad, eficiencia y disponibilidad de la información.

BIBLIOGRAFÍA

1. **GARCIA, LIC. ROSA MARIA MATO.** *DISEÑO de BASES DE DATOS.* 1999.

2. **Date, C.J.** *An introduction to database systems.* 8va. s.l. : Addison-Wesley, 2003.

3. **Maestros del web.** Maestros del web. [Online] 10 26, 2007. [Cited: 1 25, 2009.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

4. **Rodríguez Rojas, Alexeider and Pérez Pérez, Yaniel.** *Base de Datos de Gestor de Contenidos de Ajedrez.* 2008. Tesis.

5. **Marrero, María Antonia Lajús.** *Propuesta de un procedimiento para la seguridad de las bases de datos durante el diseño e implementación.* UCI. 2008. Tesis.

6. **Pereyra, Ing. Martín Tuesta.** *Sistemas de Información.* Informe.

7. **Caceres, Ana Mercedes.** [Online] 07 21, 2006. [Cited: 03 05, 2009.] Facultad de ingenieria, Universidad de Don Bosco. http://www.galeon.com/rcruz0423/docs/case.pdf.

8. **Web a Medida.** Web a Medida. [Online] 01 08, 2008. [Cited: 03 06, 2009.] Web a Medida. http://webamedida.net/index.php?option=com_content&task=view&id=1&Itemid=4.

9. **Equipo Danysoft.** Danysoft. [Online] 09 25, 2006. [Cited: 03 06, 2009.] Danysoft. Haciendo posible lo inolvidable.. http://www.danysoft.info/free/model2.pdf.

10. **González, Osmany Becerra and Martínez, Lilian Durand.** *Analisis y Diseño de un Sistema para la Generación de Reportes.* 2008. Tesis.

11. **Hernández, Carlos de la Rosa and Saquipova, Dina Yaksilik Torres.** *Análisis y Diseño de los módulos Inventario y Administracón del Proyecto ERP Cubano.* 2008. Tesis.

12. **Free Download Mnager.** FDC. [Online] 03 05, 2007. [Cited: 03 07, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5Bcuenta_de_Linux_14716_p/.

13. **Welling, Luke and Thomson, Laura.** *Desarrollo Web con PHP y MySQL.*

14. **Trueba, Visente Tejero.** Firebird en Español. *Firebird en Español.* [Online] 09 05, 2008. [Cited: 03 08, 2009.] FIREBIRD, Características básicas..

15. **León, Filein Rómmel.** [Online] 2007. [Cited: 03 08, 2009.] http://phylevn.mexrom.net/data/files/files/SQLiteFonasol.pdf.

16. **Maldonado, Daniel Martín.** Aplicaciones empresariales. [Online] 07 01, 2008. [Cited: 03 08, 2009.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

17. FCLD. [Online] 12 05, 2006. [Cited: 03 04, 2009.] http://www.codigolibre.org/modules.php?name=News&file=article&sid=4163.

18. **Espinoza, Humberto.** *PostgreSQL. Una alternativa de DBMS Open Source.* 2005.

19. **Mota, Soraya Abad.** *Entonación de Bases de Datos.* 2005.

20. **Sasha, Dennis and Bonnet, Philippe.** *Database Tuning. Principles, Experiments and Trouble Shooting Techniques.".* s.l. : Morgan Kaufmann, 2003.

21. **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico.* Quinta Edición.

22. http://teleformacion.uci.cu. [Online] 2006-2007. [Cited: 1 29, 2009.] http://teleformacion.uci.cu/mod/resource/view.php?inpopup=true&id=3374.

23. **Sánchez, Jorge.** *Diseño Conceptual de Bases de Datos.* 2004.

24. **Quintart, A.** *Méthodes d'analyse des états financiers.* 1991/1993.

25. **Eduardo.** MySQL hispano. [Online] 03 30, 2003. [Cited: 02 02, 2009.] http://www.mysql-hispano.org/page.php?id=16&pag=1.

26. **González Díaz-Bethencourt, Jose Luis.** *Sistemas Gestores de Bases de Datos.* pp. 18-47.

27. **Espinoza, Humberto.** *PostgreSQL Una Alternativa de DBMS Open Source.* 2005.

28. **Elmasri, R. and Navathe, S.** *Sistemas de bases de datos.* s.l. : Prentice-Hall, 2002. p. 955. 84-7829-051-6.

29. **Elmasri, A. and Navathe, S.** *Fundamentos de Sistemas de Bases de Datos.* 3ra. s.l. : Addison-Wesley, 2002. 84-782-9051-6.

30. **Piattini, M., et al.** *Tecnología y Diseño de Bases de Datos.* s.l. : RA_MA, 2006. p. 980. 8478977333.

31. **Vázquez Martínez, Salvador and Martínez Alcántara2, Antonio F.** *UNA ARQUITECTURA PARA EL ANÁLISIS AUTOMATIZADO DE BASES DE DATOS.* 2003. pp. 89-106. Vol. 7.

32. **Piattini, M. and Díaz, O.** *Advanced database technology and design.* s.l. : Artech House Publishers, 2000. p. 535. 0-89006-395-8.

33. **Date, C.J.** *An introduction of database systems.* 7ma. s.l. : Prentice-Hall, 2001. p. 936. 968-444-419-2.

34. **Connolly, T. and Begg, C.** *Database Systems.* 3ra. s.l. : Addison-Wesley, 2002. p. 1236. 0-201-70857-4.

35. **Atzeni, P., et al.** *Database Systems. Concepts, languages and architectures.* s.l. : McGraw-Hill, 1999. p. 612. 0-07-709500-6.

36. **Batini, Carlo, Ceri, Stefano and Navathe, Shamkant B.** *Diseño conceptual de bases de datos.* s.l. : Addison-Wesley / Díaz de Santos, 1991. 0201601206.

37. **Silberschatz, A., Korth, H.F and Sudarshan, S.** *Fundamentos de bases de datos.* 4ta. s.l. : McGraw-Hill, 2002. p. 787. 84-481-3654-3.

38. **Garrido Picazo, Piedad, Coll Villalta, Manuel and Mártinez Domínguez, Francisco J.** *DerEditor4GL: Software para la docencia en el diseño de Bases.*

39. **PostgreSQL.** PostgreSQL. [Online] [Cited: 03 08, 2009.] Pagina del SGBD PostgreSQL. http://www.postgresql.org/.

40. **Centro de Capacitación en Informática. Montevideo - Uruguay.** Centro de Capacitación en Informática. [Online] [Cited: 03 09, 2009.] http://asp.anep.edu.uy/capinfo/Material/Access/Acc_cap13.pdf.

ANEXOS

Anexo # 1: Componente Persona

Figura 1: Modelo de datos del componente Persona

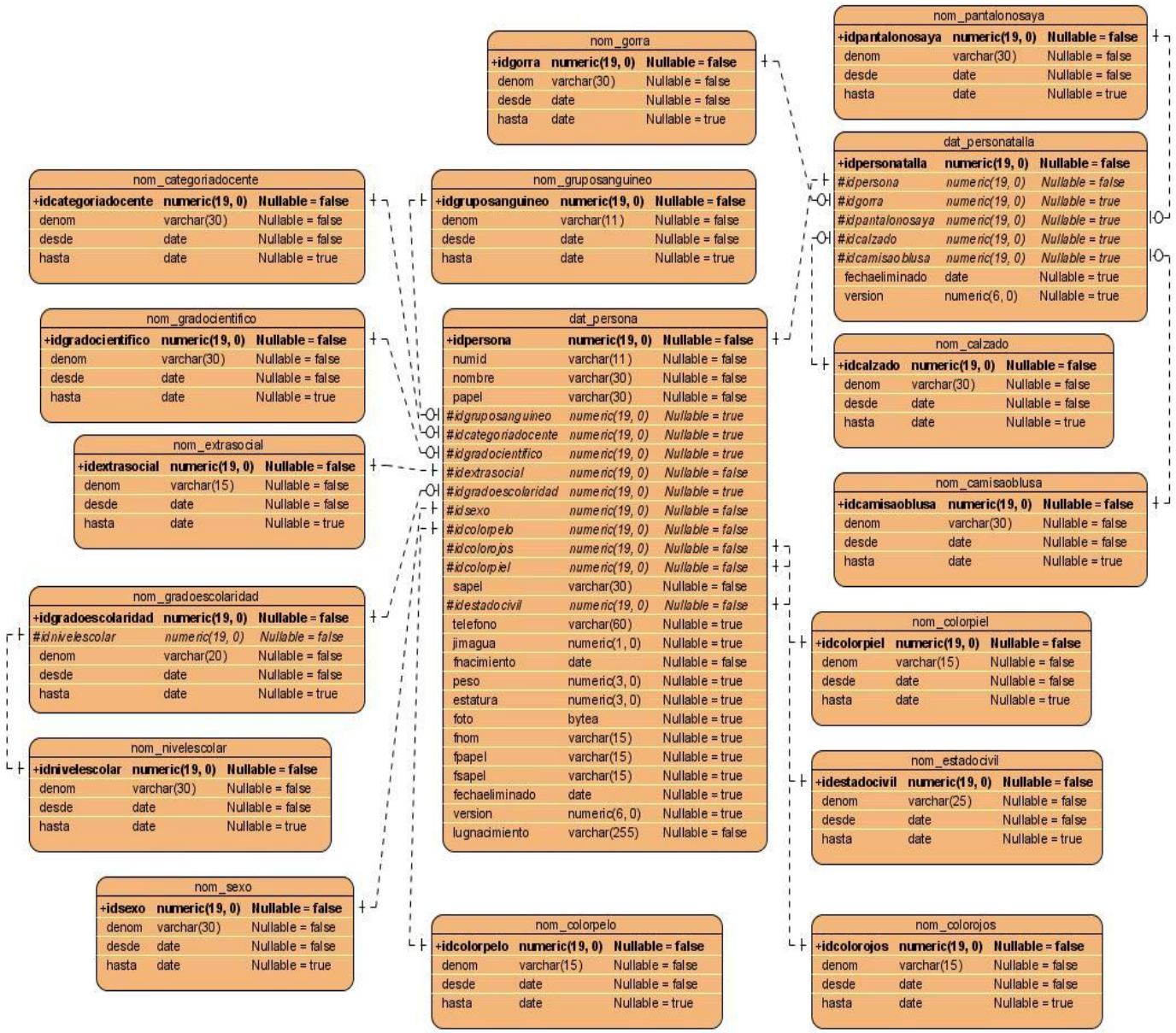


Tabla 2: Descripción de las tablas del componente Persona

Tablas	Descripción
nom_gorra	Almacena las diferentes tallas existentes de gorra.
nom_pantalonosaya	Almacena las diferentes tayas de pantalón o saya.
nom_calzado	Almacena los diferentes números de calzado.
nom_camisaoblusa	Almacena las diferentes tallas existentes de camisa o blusa.
nom_gruposanguineo	Almacenan los grupos sanguíneos conocidos hasta el momento.
nom_categoriadocente	Almacena las categorías docentes que existen.
nom_gradocientifico	Almacena los grados científicos que puede tener una persona.
nom_extrasocial	Almacena los diferentes tipos de procedencia social que existen.
nom_nivelescolar	Almacena los diferentes niveles escolares que puede tener una persona.
nom_sexo	Almacena los sexos existentes.
nom_colorpelo	Almacena los diferentes colores de pelo.
nom_colorojos	Almacena los diferentes colores de ojos.
nom_colorpiel	Almacena los diferentes colores de piel.
nom_estadocivil	Almacena los estados civiles que puede poseer la persona.

Tabla 3: Descripción de la tabla dat_persona

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idpersona	numeric(19)	PK	No	Identificador de la persona
numid	varchar(11)		No	Carnet de identidad de la persona
nombre	varchar(30)		No	Nombre de la persona

papel	varchar(30)		No	Primer apellido de la persona
sapel	varchar(30)		No	Segundo apellido de la persona
lugnacimiento	varchar(255)		No	Lugar de nacimiento de la persona
idgruposanguineo	numeric(19)	FK	Si	Identificador del grupo sanguíneo
idcategroriadocente	numeric(19)	PK	Si	Identificador de la categoría docente
idgradocientifico	numeric(19)	PK	Si	Identificador del grado científico
idextrasocial	numeric(19)	PK	No	Identificador de la procedencia social
idgradoescolaridad	numeric(19)	PK	Si	Identificador del grado de escolaridad
idsexo	numeric(19)	PK	No	Identificador del sexo
idcolorpelo	numeric(19)	PK	No	identificador del color de pelo
idcolorojos	numeric(19)	PK	No	Identificador del color de ojos
idcolorpiel	numeric(19)	PK	No	Identificador del color de piel
idestadocivil	numeric(19)	PK	No	Identificador del estado civil
teléfono	varchar(60)		Si	Teléfono de la persona(si tiene)
jimagua	numeric(1)		Si	Si la persona es jimagua o no(1- si lo es, 0- si no lo es)
fnacimiento	date		No	Fecha de nacimiento de la persona
peso	numeric(3)		Si	Pero de la persona
estatura	numeric(3)		Si	Estatura de la persona
foto	bytea		Si	Foto de la persona
fnom	varchar(15)		Si	Fonética del nombre

fpapel	varchar(15)		Si	Fonética del primer apellido
fsapel	varchar(15)		Si	Fonética del segundo apellido
fechaeliminado	date		Si	Se utiliza para no eliminar físicamente a la persona de la base de datos
version	numeric(6)		Si	Se utiliza para la concurrencia

Tabla 4: Descripción de la tabla dat_personatalla

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idpersonatalla	numeric(19)	PK	No	Identificador de la talla
idpersona	numeric(19)	FK	No	Identificador de la persona
idgorra	numeric(19)	FK	Si	Identificador de la gorra
idpantalonosaya	numeric(19)	FK	Si	Identificador del pantalón o la saya
idcalzado	numeric(19)	FK	Si	Identificador del calzado
idcamisaoblusa	numeric(19)	FK	Si	Identificador de la camisa o blusa
fechaeliminado	date		No	Se utiliza para no eliminarla físicamente de la base de datos
versión	numeric(6)		Si	Se utiliza para la concurrencia

Tabla 5: Descripción de la tabla nom_gradoescolaridad

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idgradoescolaridad	numeric(19)	PK	No	Identificador de las tallas de las camisa o blusa
idnivelescolar	numeric(19)	FK	No	Identificador del nivel escolar
denom	varchar(20)		No	Descripción del grado de escolaridad

desde	date		No	Fecha en la que se inserta el grado de escolaridad
hasta	date		Si	Fecha en la que se elimina el grado de escolaridad

Tabla 6: Descripción de la tabla nom_gorra

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idgorra	numeric(19)	PK	No	Identificador de la talla de la gorra
denom	varchar(30)		No	Descripción de la talla de la gorra
desde	date		No	Fecha en la que se inserta la talla de la gorra
hasta	date		Si	Fecha en que se elimina la talla de la gorra

Tabla 7: Descripción de la tabla nom_pantalonosaya

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idgorra	numeric(19)	PK	No	Identificador del pantalón o la saya
denom	varchar(30)		No	Descripción del pantalón o la saya
desde	date		No	Fecha en la que se inserta el pantalón o la saya
hasta	date		Si	Fecha en que se elimina el pantalón o la saya

Tabla 8: Descripción de la tabla nom_calzado

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcalzado	numeric(19)	PK	No	Identificador del calzado
denom	varchar(30)		No	Descripción del calzado

desde	date		No	Fecha en la que se inserta el calzado
hasta	date		Si	Fecha en que se elimina el calzado

Tabla 9: Descripción de la tabla nom_camisaoblusa

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcamisaoblusa	numeric(19)	PK	No	Identificador de las tallas de las camisa o blusa
denom	varchar(30)		No	Descripción del número de las tallas de camisa o blusa
desde	date		No	Fecha en la que se inserta la camisa o blusa
hasta	date		Si	Fecha en la que la camisa o blusa deja de existir

Tabla 10: Descripción de la tabla nom_gruposanguineo

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idgruposanguineo	numeric(19)	PK	No	Identificador del grupo sanguíneo
denom	varchar(11)		No	Descripción del grupo sanguíneo
desde	date		No	Fecha en la que se inserta el grupo sanguíneo
hasta	date		Si	Fecha en que se elimina el grupo sanguíneo

Tabla 11: Descripción de la tabla nom_categoriadocente

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcategoriadocente	numeric(19)	PK	No	Identificador de la categoría docente
denom	varchar(30)		No	Descripción de la categoría docente

desde	date		No	Fecha en la que se inserta la categoría docente
hasta	date		Si	Fecha en que se elimina la categoría docente

Tabla 12: Descripción de la tabla nom_gradocientifico

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idgradocientifico	numeric(19)	PK	No	Identificador del grado científico
denom	varchar(30)		No	Descripción del grado científico
desde	date		No	Fecha en la que se inserta el grado científico
hasta	date		Si	Fecha en que se elimina el grado científico

Tabla 13: Descripción de la tabla nom_extrasocial

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idextrasocial	numeric(19)	PK	No	Identificador de la procedencia social
denom	varchar(15)		No	Descripción de la procedencia social
desde	date		No	Fecha en la que se inserta la procedencia social
hasta	date		Si	Fecha en que se elimina la procedencia social

Tabla 14: Descripción de la tabla nom_nivelescolar

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idnivelescolar	numeric(19)	PK	No	Identificador del nivel escolar
denom	varchar(30)		No	Descripción del nivel escolar
desde	date		No	Fecha en la que se inserta el nivel escolar

hasta	date		Si	Fecha en la que se elimine el nivel escolar

Tabla 15: Descripción de la tabla nom_sexo

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idsexo	numeric(19)	PK	No	Identificador del sexo
denom	varchar(30)		No	Descripción del sexo
desde	date		No	Fecha en la que se inserta el sexo
hasta	date		Si	Fecha en la que se elimine el sexo

Tabla 16: Descripción de la tabla nom_colorpelo

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcolorpelo	numeric(19)	PK	No	Identificador del color de pelo
denom	varchar(15)		No	Descripción del color de pelo
desde	date		No	Fecha en la que se inserta el color de pelo
hasta	date		Si	Fecha en la que el color de pelo deja de existir

Tabla 17: Descripción de la tabla nom_colorojos

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcolorojos	numeric(19)	PK	No	Identificador del color de ojo
denom	varchar(15)		No	Descripción del color de ojo
desde	date		No	Fecha en la que se inserta el color de ojo
hasta	date		Si	Fecha en la que el color de ojo deja de existir

Tabla 18: Descripción de la tabla nom_colorpiel

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcolorpiel	numeric(19)	PK	No	Identificador del color de piel
denom	varchar(15)		No	Descripción del color de piel
desde	date		No	Fecha en la que se inserta el color de piel
hasta	date		Si	Fecha en la que el color de piel deja de existir

Tabla 19: Descripción de la tabla nom_estadocivil

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idestadocivil	numeric(19)	PK	No	Identificador del estado civil
denom	varchar(25)		No	Descripción del estado civil
desde	date		No	Fecha en la que se inserta el estado civil
hasta	date		Si	Fecha en la que el estado civil deja de existir

Anexo # 2: Componente Trabajador

Figura 2: Modelo de Datos del componente Trabajadores

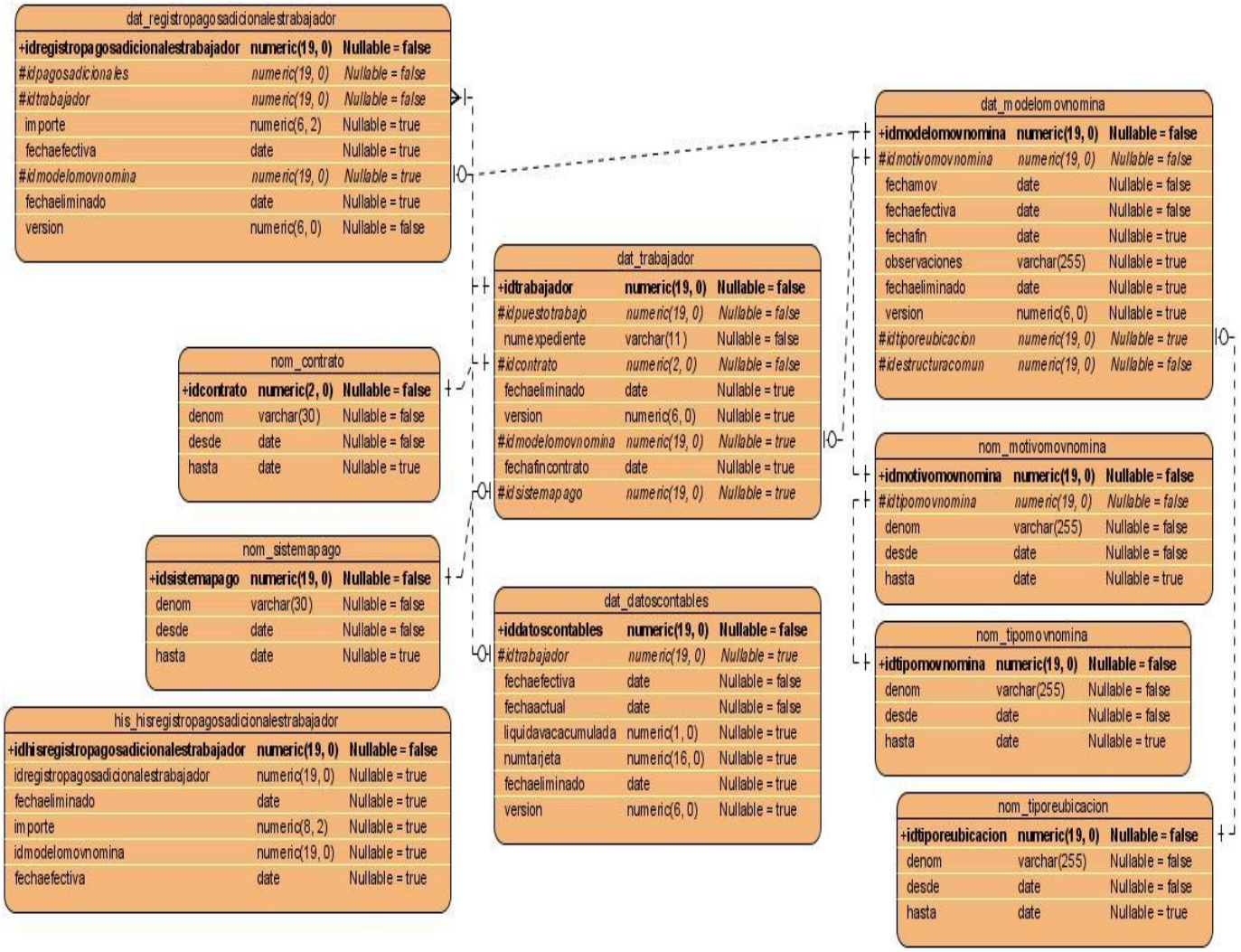


Tabla 20: Descripción de las tablas del componente Trabajador

Tablas	Descripción
dat_registropagosadicionalestrabajador	Almacena los pagos adicionales asociados al Trabajador.
nom_sistemapago	Almacena los tipos de pagos existentes.
nom_contrato	Almacena los diferentes tipos de contrato que puede tener un trabajador.
nom_motivomovnomina	Almacena los motivos de movimiento de Nómina.
nom_tipomovnomina	Almacena los diferentes tipo de Nómina.
nom_tiporeubicacion	Almacena los tipos de reubicación que puede tener un trabajador.
his_hisregistropagosadicionalestrabajador	Almacena una historial de los Pagos adicionales asociados del trabajador.

Tabla 21: Descripción de la Tabla dat_trabajador

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtrabajador	numeric(19)	PK	No	Identificador del Trabajador
idpuestotrabajo	numeric(19)	FK	No	Identificador del puesto de trabajo que ocupa el trabajador
numexpediente	varchar(11)		No	Número del expediente del trabajador
idcontrato	numeric(19)	FK	No	Identificador del contrato del trabajador
fechaeliminado	date		Si	Se utiliza para no eliminar al trabajador físicamente de la BD.
version	numeric(6)		Si	Se utiliza para la concurrencia
idimpuesto	numeric(19)	FK	Si	Identificador del impuesto que se le cobra al trabajador para la seguridad social
idmodelomovnomina	numeric(19)	FK	Si	Identificador del modelo de movimiento de nómina que se crea cuando se crea el

				trabajador
fechafincontrato	date		Si	Fecha de finalización del contrato del trabajador si es que tiene
idsistemapago	numeric(19)	FK	Si	Identificador del sistema de pago que se asigna al trabajador

Tabla 22: Descripción de la Tabla dat_modelomovnomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idmodelomovnomina	numeric(19)	PK	No	Identificador del modelo de movimiento de nómina
idmotivomovnomina	numeric(19)	FK	No	Identificador del motivo de movimiento de nómina
fechamov	date		No	Fecha en que se realiza el movimiento
fechaefectiva	date		No	Fecha a partir de la cual se hará vigente el movimiento
fechafin	date		Si	Fecha en la cual termina el movimiento
observaciones	varchar(255)		Si	Observaciones del movimiento de nómina
fechaeliminado	date		Si	Se utiliza para no eliminar el movimiento físicamente de la BD
version	numeric(6)		Si	Se utiliza para la concurrencia
idtiporeubicacion	numeric(19)	FK	Si	Identificador del tipo de reubicación
idestructuracomun	numeric(19)	FK	Si	Identificador de la entidad a la que pertenece el movimiento de nómina

Tabla 23: Descripción de la Tabla dat_datoscontables

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
iddatoscontables	numeric(19)	PK	No	Identificador de los datos contables
idtrabajador	numeric(19)	FK	No	Identificador del trabajador al que pertenecen los datos
idcuenta	numeric(19)	FK	No	Identificador de la cuenta
fechaefectiva	date		No	Fecha a partir de la cual va a ser efectivo
fechaactual	date		No	Fecha actual del sistema
liquidavacacumulada	numeric(1)		Si	Si liquida vacaciones acumuladas o no (0- si no liquida, 1- si liquida)
idcentrocomun	numeric(19)	FK	No	Identificador del centro de costo
numtarjeta	numeric(16)		Si	Numero de tarjera del trabajador
fechaeliminado	date		Si	Se utiliza para no eliminar los datos físicamente de la BD
version	numeric(6)		Si	Se utiliza para la concurrencia

Tabla 24: Descripción de la Tabla dat_registropagosadicionalestrabajador

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idregistropagosadicionalestrabajador	numeric(19)	PK	No	Identificador del registro de pagos adicionales del trabajador
idpagosadicionales	numeric(19)	FK	No	Identificador del pago adicional
idtrabajador	numeric(19)	FK	No	Identificador del trabajador
importe	numeric(6)		Si	Importe del pago adicional asociado al trabajador
fechaefectiva	date		Si	Fecha en la que se hace efectivo el cambio efectuado en el registro

				de pagos adicionales
idmodelomovnomina	numeric(19)	FK	Si	Identificador del modelo de movimiento de nómina que se crea cuando se modifica el pago adicional
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD
version	numeric(6)		No	Se utiliza para la concurrencia

Tabla 25: Descripción de la Tabla nom_sistemapago

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idsistemapago	numeric(19)	PK	No	Identificador del sistema de pago
denom	varchar(30)		No	Descripción del sistema de pago
desde	date		No	Fecha en la que se inserta el sistema de pago
hasta	date		Si	Fecha en la que se elimina el sistema de pago

Tabla 26: Descripción de la Tabla nom_contrato

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcontrato	numeric(2)	PK	No	Identificador del contrato
denom	varchar(30)		No	Descripción del contrato
desde	date		No	Fecha en la que se inserta el contrato
hasta	date		Si	Fecha en la que se elimina el contrato

Tabla 27: Descripción de la Tabla nom_motivomovnomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idmotivomovnomina	numeric(19)	PK	No	Identificador del motivo de movimiento de nómina
idtipomovnomina	numeric(19)	FK	No	Identificador del tipo de movimiento de nómina
denom	varchar(30)		No	Descripción del motivo de movimiento de nómina
desde	date		No	Fecha en la que se inserta el motivo de movimiento de nómina
hasta	date		Si	Fecha en la que se elimina el motivo de movimiento de nómina

Tabla 28: Descripción de la Tabla nom_tiponomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtipomovnomina	numeric(19)	PK	No	Identificador del tipo de movimiento de nómina
denom	varchar(255)		No	Descripción del tipo de movimiento de nómina
desde	date		No	Fecha en la que se inserta el tipo de movimiento de nómina
hasta	date		Si	Fecha en la que se elimina el tipo de movimiento de nómina

Tabla 29: Descripción de la Tabla nom_tiporeubicacion

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtiporeubicacion	numeric(19)	PK	No	Identificador del tipo de reubicación
denom	varchar(255)		No	Descripción del tipo de reubicación
desde	date		No	Fecha en la que se inserta el tipo de

				reubicación
hasta	date		Si	Fecha en la que se elimina el tipo de reubicación

Tabla 30: Descripción de la Tabla his_hisregistropagosadicionalestrabajador

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idhisregistropagosadicionalestrabajador	numeric(19)		No	Identificador del Histórico de Registro de Pagos Adicionales asociados a un Trabajador
idregistropagosadicionalestrabajador	numeric(19)		Si	Identificador del registro de pagos adicionales del trabajador
fechaeliminado	date		Si	Se utiliza para no eliminar físicamente de la BD
importe	numeric(8)		Si	Importe del pago adicional
idmodelomovnomina	numeric(19)		Si	Identificador del modelo de movimiento de nómina
fechaefectiva	date		Si	Fecha en la que se hace efectiva el pago adicional

Anexo # 3: Componente Pagos Adicionales

Figura 3: Modelo de Datos del componente Pagos Adicionales

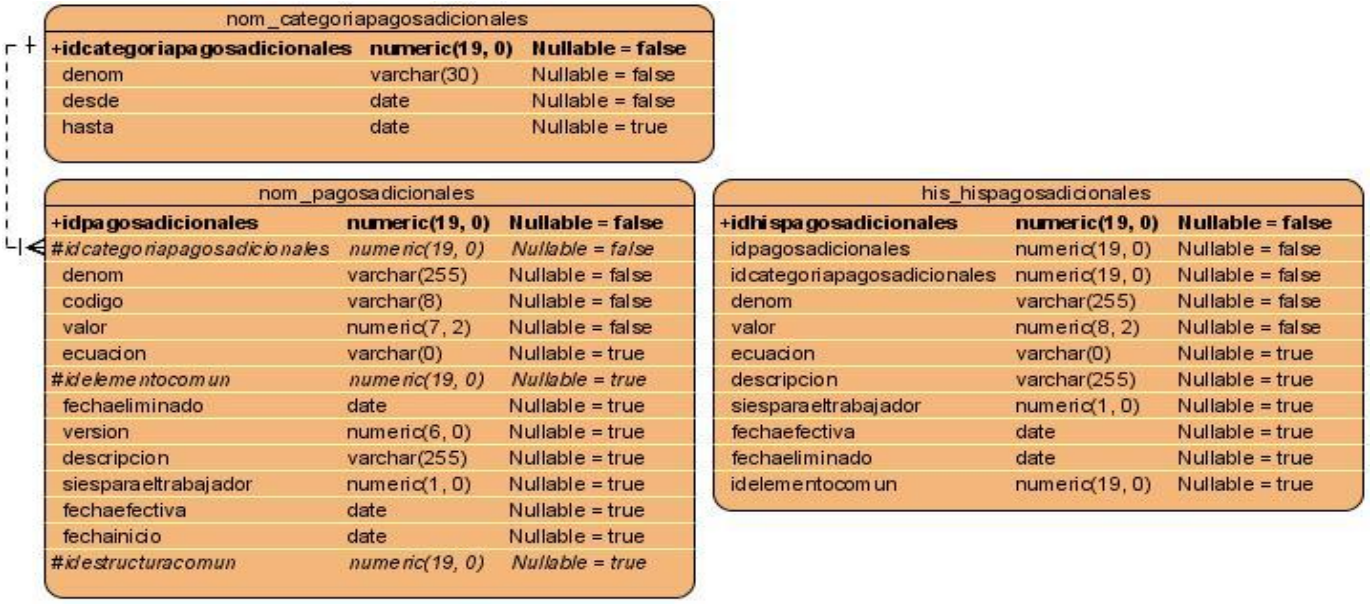


Tabla 31: Descripción de la Tabla nom_pagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idpagosadicionales	numeric(19)	PK	No	Identificador del pago adicional
idcategoriapagosadicionales	numeric(19)	FK	No	Identificador de la categoría del pago adicional
denom	varchar(255)		No	Denominación del pago adicional
codigo	varchar(8)		No	Código del pago adicional
valor	numeric(7)		No	Equivalente al pago del trabajador
ecuacion	varchar(255)		Si	Ecuación para calcular el salario del trabajador

idelementocomun	numeric(19)	FK	Si	identificador del elemento común
fechaeliminado	date		Si	Fecha en la que se elimina el pago adicional
version	numeric(6)		No	Utilizado para la concurrencia
descripcion	varchar(255)		Si	Descripción del pago adicional
siesparaeltrabajador	numeric(1)		No	0- para el trabajador, 1- para el puesto de trabajo
fechaefectiva	date		Si	Fecha en que se hace efectiva el pago
fechainicio	date		Si	Fecha en que se registra el pago adicional
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 32: Descripción de la Tabla nom_categoriapagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcategoriapagosadicionales	numeric(19)	PK	No	Identificador de la categoría de pagos adicionales
denom	varchar(30)		No	Descripción de la categoría
desde	date		No	Fecha en la que se inserta la categoría
hasta	date		Si	Fecha en la que se elimina la categoría

Tabla 33: Descripción de la Tabla his_hispagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idhispagosadicionales	numeric(19)	PK	No	Identificador del historial del pago adicional

idpagosadicionales	numeric(19)		No	Identificador del pago adicional
idcategoriapagosadicionales	numeric(19)		No	Identificador de la categoría del pago adicional
denom	varchar(255)		No	Denominación del pago adicional
valor	numeric(8)		No	Valor del pago adicional asignado al trabajador
ecuacion	varchar(0)		Si	Ecuación para calcular el salario de un trabajador
descripcion	varchar(255)		Si	descripción del pago adicional
siesparaeltrabajador	numeric(1)		Si	Indica si el pago adicional es o no para el trabajador
fechaefectiva	date		Si	Fecha en la que se hace efectiva el pago adicional
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD
idelementocomun	numeric(19)		Si	Identificador del elemento de gasto

Anexo # 4: Componente Puesto de Trabajo

Figura 4: Modelo de Datos del componente Puesto de Trabajo

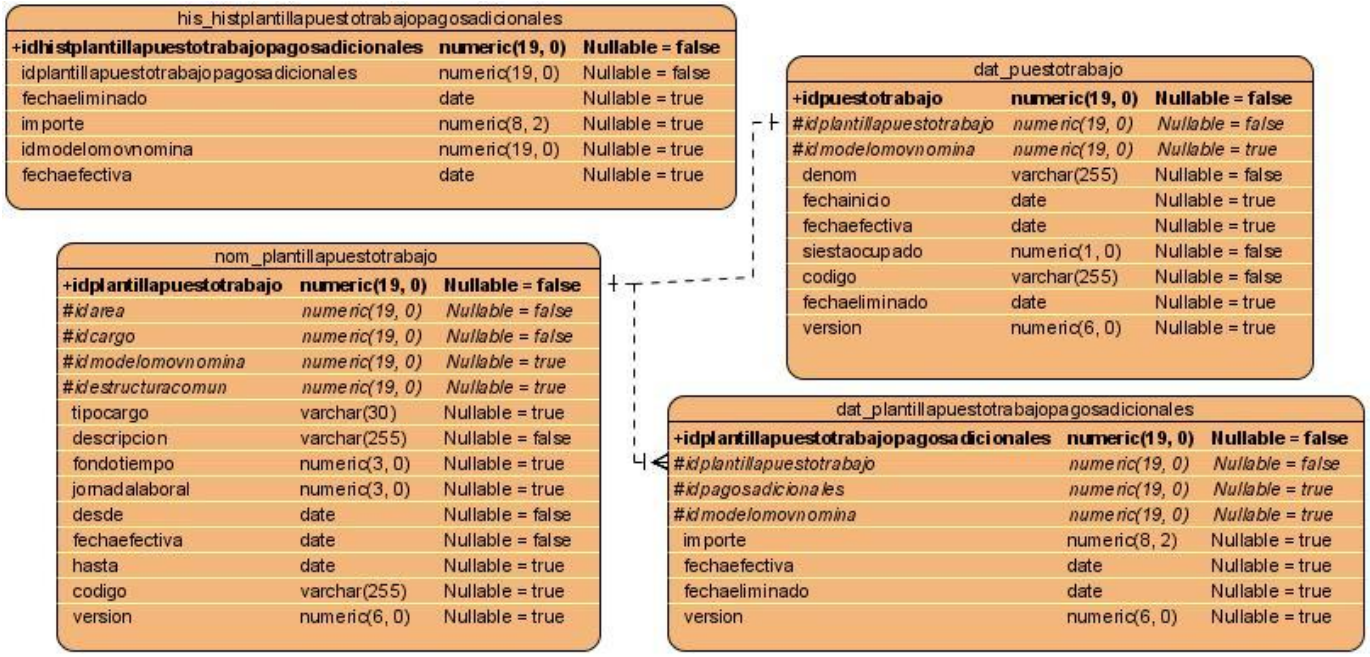


Tabla 34: Descripción de la Tabla dat_puestotrabajo

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idpuestotrabajo	numeric(19)	PK	No	Identificador del puesto de trabajo
idplantillapuestotrabajo	numeric(19)	FK	No	Identificador de la plantilla de puesto de trabajo
denom	varchar(255)		No	Denominación del puesto de trabajo
fechainicio	date		Si	Fecha en la que se crea el puesto de trabajo
fechaefectiva	date		Si	Fecha en la que comienza a ser efectivo el puesto de trabajo

siestaocupado	numeric(1)		No	Indica si el puesto de trabajo esta o no ocupado (1- si esta ocupado, 0- si no)
codigo	varchar(255)		No	Código del puesto de trabajo
idmodelomovnomina	numeric(19)	FK	Si	Identificador del modelo de movimiento de nómina
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD
version	numeric(6)		Si	Se utiliza para la concurrencia

Tabla 35: Descripción de la Tabla dat_plantillapuestotrabajopagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idplantillapuestotrabajopagosadicionales	numeric(19)	PK	No	Identificador de la plantilla de puesto de trabajo de pagos adicionales
idplantillapuestotrabajo	numeric(19)	FK	No	Identificador de la plantilla de puesto de trabajo
idpagosadicionales	numeric(19)	FK	No	Identificador del pago adicional
importe	numeric(8)		Si	Importe del pago adicional asociado a la plantilla de puesto de trabajo
fechaefectiva	date		Si	Fecha a partir de la cual se hacen efectivos los cambios efectuados sobre la plantilla de puesto de trabajo
idmodelomovnomina	numeric(19)	FK	Si	Identificador del modelo de movimiento de nómina
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD

version	numeric(6)		Si	Para la concurrencia

Tabla 36: Descripción de la Tabla nom_plantillapuestotrabajo

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idplantillapuestotrabajo	numeric(19)	PK	No	Identificador de plantilla de puestos de trabajos
idcargo	numeric(19	FK	No	Identificador del cargo
tipocargo	varchar(30)		Si	Si el cargo es militar o civil
descripcion	varchar(255)		No	Descripción de la plantilla de puestos de trabajos
fondotiempo	numeric(3)		Si	Fondo de tiempo de trabajo
jornadalaboral	numeric(3)		Si	Jornada laboral que debe trabajar
idarea	numeric(19)	FK	No	identificador del área
desde	date		No	Fecha en que se inserta la plantilla de puestos de trabajos
fechaefectiva	date		No	Fecha en que entra en vigor
hasta	date		Si	Fecha en que se elimina la plantilla de puestos de trabajos
idmodelomovnomina	numeric(19)	FK	Si	Identificador del modelo de movimiento de nómina
codigo	varchar(255)		Si	Código de la plantilla de puestos de trabajos
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad
versión	numeric(6)		No	Para la concurrencia

Tabla 37: Descripción de la Tabla his_hisplantillapuestotrabajopagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idhisplantillapuestotrabajopagosadicionales	numeric(19)	PK	No	Identificador del historial de plantilla de puesto de trabajos
idplantillapuestotrabajopagosadicionales	numeric(19)		No	Identificador de la plantilla de puesto de trabajo con pagos adicionales asociados
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD
importe	numeric(8)		Si	importe del pago adicional
idmodelomovnomina	numeric(19)		Si	Identificador del modelo de movimiento de nómina
fechaefectiva	date		Si	Fecha en que se hace efectiva la modificación en la Plantilla de Puesto de Trabajo

Anexo # 5: Componente Incidencia

Figura 5: Modelo de Datos del componente Incidencia

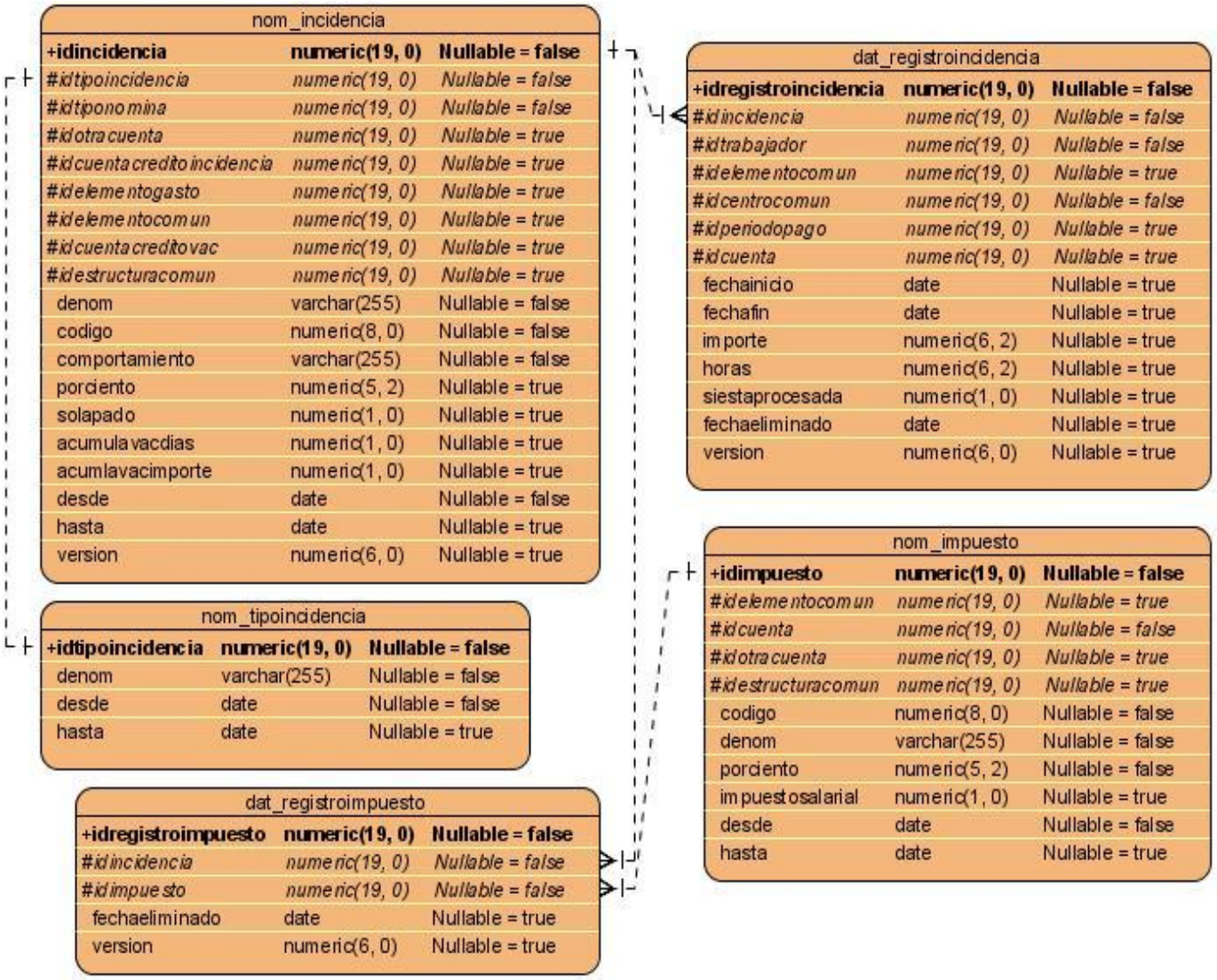


Tabla 38: Descripción de la tabla dat_registroincidencia

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idregistroincidencia	numeric(19)	PK	No	Identificador del registro de incidencia
idtrabajador	numeric(19)	FK	No	Identificador del trabajador
idincidencia	numeric(19)	FK	No	Identificador de la incidencia
fechainicio	date		Si	Fecha en la que se elimina la incidencia
fechafin	date		Si	Fecha en la que se termina la incidencia
importe	date		Si	Importe de la incidencia
idelementocomun	numeric(19)	FK	Si	Identificador del elemento de gasto
idcentrocomun	numeric(19)	FK	No	Identificador del centro de costo
horas	numeric(6)		Si	Horas reportadas en la incidencia
siestaprocesada	numeric(1)		Si	Indica si la Incidencia esta o no procesada ejemplo (1 procesada, 0 no procesada)
fechaeliminado	date		Si	Se utiliza para no eliminarla físicamente de la BD
idperiodopago	numeric(19)	FK	No	identificador del periodo de pago
version	numeric(6)		Si	Se utiliza para la concurrencia
idcuenta	numeric(19)	FK	No	Identificador de la cuenta

Tabla 39: Descripción de la Tabla dat_resgistroimpuesto

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idregistroimpuesto	numeric(19)	PK	No	Identificador del registro de impuesto
idimpuesto	numeric(19)	FK	No	Identificador del impuesto
idincidencia	numeric(19)	FK	No	identificador de la incidencia

fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la BD
version	numeric(6)		Si	Se utiliza para la concurrencia

Tabla 40: Descripción de la Tabla nom_incidencia

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idincidencia	numeric(19)	PK	No	Identificador de la incidencia
idtipoincidencia	numeric(19)	FK	No	Identificador del tipo de incidencia
idtiponomina	numeric(19)	FK	No	Identificador del tipo de nómina
idotracuenta	numeric(19)	FK	Si	Identificador de cuenta
idcuentacreditoincidencia	numeric(19)	FK	Si	Identificador de cuenta
idelementogasto	numeric(19)	FK	Si	Identificador del elemento de gasto
idelementocomun	numeric(19)	FK	Si	
idcuentacreditovac	numeric(19)	FK	Si	Identificador de cuenta
denom	varchar(255)		No	Denominación de la incidencia
codigo	numeric(8)		No	Código de la incidencia
comportamiento	varchar(255)		No	Comportamiento de la incidencia
porciento	numeric(5)		Si	Porciento en que afecta la incidencia en el salario del trabajador
solapado	numeric(1)		Si	Determina si puedo o no ser concurrente con otra incidencia (0- no, 1- si)
acumulavacdias	numeric(1)		Si	Indica si acumula vacaciones en días(1- no acumula, 2- si acumula)
acumulavacimporte	numeric(1)		Si	0- no acumula, 1- acumula, 2- acumula y liquida

desde	date		No	Fecha de inicio de la incidencia
hasta	date		Si	Fecha en que se elimina la incidencia
version	numeric(6)		Si	Se utiliza para la concurrencia
idestructuracomun	numeric(19)	FK	Si	Identificador de la entidad

Tabla 41: Descripción de la tabla nom_impuesto

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idimpuesto	numeric(19)	PK	No	Identificador del impuesto
codigo	numeric(8)		No	Código del impuesto
denom	varchar(255)		No	Denominación del impuesto
porciento	numeric(5)		No	Porciento del impuesto
idelementocomun	numeric(19)	FK	Si	
idcuenta	numeric(19)	FK	No	Identificador de la cuenta
idotracuenta	numeric(19)	FK	Si	Identificador de otra cuenta
impuestosalarial	numeric(1)		Si	Si el impuesto es salarial o no (1- si es salarial, 0- si no lo es)
desde	date		No	Fecha inicial del registro de la incidencia
hasta	date		Si	Fecha fin de vigencia del impuesto
idestructuracomun	numeric(19)	FK	Si	Identificador de la entidad

Tabla 42: Descripción de la Tabla nom_tipoincidencia

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtipoincidencia	numeric(19)	PK	No	Identificador del tipo de incidencia
denom	varchar(255)		No	Descripción del tipo de incidencia

desde	date		No	Fecha en la que se inserta el tipo de incidencia
hasta	date		Si	Fecha en la que se elimina el tipo de incidencia

Anexo # 6: Componente Nomina

Figura 6: Modelo de Datos del componente Nomina

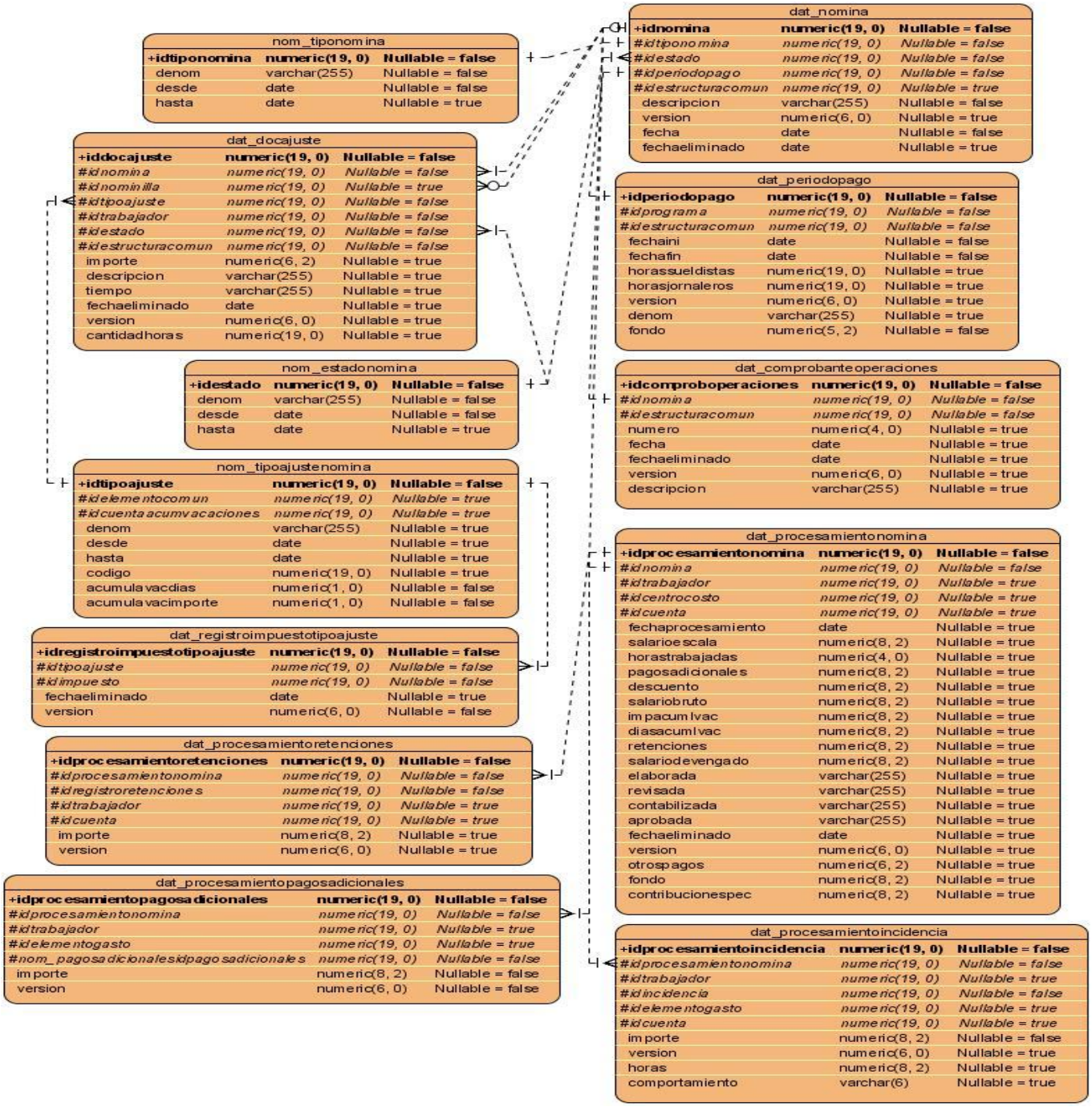


Tabla 43: Descripción de las tablas del componente Nomina

Tablas	Descripción
dat_registroimpuestotipoajuste	Es la asociación del Registro de impuesto con el Tipo de ajuste.
dat_comprobanteoperaciones	Almacena los comprobantes de operaciones que se realicen.
dat_procesamientopagosadicionales	Almacena el resultada del calculo de los pagos adicionales de cada trabajador de la nómina.
dat_procesamientoincidencia	Almacena el cálculo de las incidencias de cada trabajador de la nómina.
dat_procesamientoretenciones	Almacena el resultado del procesamiento de las retenciones.
nom_tipoajustenomina	Almacena los Tipos de ajuste que se le pueden realizar a la nómina.
nom_tiponomina	Almacena los diferentes tipos de nómina que existen.
nom_estadonomina	Almacena los diferentes estados en los que se puede encontrar la nómina.

Tabla 44: Descripción de la Tabla dat_nomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idnomina	numeric(19)	PK	No	Identificador de la nómina
idtiponomina	numeric(19)	FK	No	Identificador del tipo de nómina
idperiodopago	numeric(19)	FK	No	Identificador del periodo de pago
idestado	numeric(19)	FK	No	Identificador del estado
version	numeric(6)		Si	Se utiliza para la concurrencia
fecha	date		No	Fecha en la que se emite la nómina
descripción	varchar(255)		No	Descripción de la nómina

idestructuracomun	numeric(19)	FK	Si	Identificador de la entidad
fechaeliminado	date		No	Se utiliza para no eliminarla físicamente de la base de datos

Tabla 45: Descripción de la Tabla dat_periodopago

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idperiodopago	numeric(19)	PK	No	Identificador del periodo de pago
idprograma	numeric(19)		No	Identificador del programa
fechaini	date		No	Fecha de inicio del periodo de pago
fechafin	date		No	Fecha de fin del periodo de pago
horassueldistas	numeric(19)		Si	Cantidad de horas para sueldistas
horasjornaleros	numeric(19)		Si	Cantidad de horas para jornaleros
version	numeric(6)		Si	Se utiliza para la concurrencia
denom	varchar(255)		No	Denominación del pago adicional
fondo	numeric(5)		No	Porciento de sueldo y pago adicional a pagar según el periodo de pago
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 46: Descripción de la Tabla dat_docajuste

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
iddocajuste	numeric(19)	PK	No	Identificador del documento de ajuste
idtipoajuste	numeric(19)	FK	No	Identificador del tipo de ajuste
idtrabajador	numeric(19)	FK	No	Identificador del trabajador
idnomina	numeric(19)	FK	No	Identificador de la nómina

idestado	numeric(6)	FK	No	Identificador del estado
importe	numeric(6)		Si	Importe a corregir en el documento de ajuste
descripción	varchar(255)		Si	Descripción del documento de ajuste
tiempo	varchar(255)		Si	Tiempo a corregir en el documento de ajuste
fechaeliminado	date		Si	Se usa para no eliminarlo físicamente de la base de datos
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad
versión	numeric(6)		Si	Se utiliza para la concurrencia
idnominilla	numeric(19)	FK	Si	Identificador de la nómina que esta procesada
cantidadhoras	numeric(19)		Si	Cantidad de horas a corregir en el documento de ajuste

Tabla 47: Descripción de la Tabla dat_procesamientonomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idprocesamientonomina	numeric(19)	PK	No	Identificador del procesamiento de nómina
idnomina	numeric(19)	FK	No	Identificador de la nómina
idtrabajador	numeric(19)	PK	Si	Identificador del trabajador
fechaprocesamiento	date		Si	Fecha en que se procesa la nómina
salarioescala	numeric(8)		Si	Salario escala
horastrabajadas	numeric(4)		Si	Horas trabajadas
pagosadicionales	numeric(8)		Si	Total del importe de los pagos adicionales del trabajador

descuento	numeric(8)		Si	Descuento en importe que se le hacen al trabajador (por ejemplo las penalizaciones)
salariobruto	numeric(8)		Si	Salario bruto del trabajador
impacumlvac	numeric(8)		Si	Importe acumulado de vacaciones
diasacumlvac	numeric(8)		Si	Importe acumulado en días
retenciones	numeric(8)		Si	Retenciones reportadas al trabajador
salariodevengado	numeric(8)		Si	Salario devengado del trabajador
elaborada	varchar(255)		Si	Usuario que elabora la nómina
revisado	varchar(255)		Si	Usuario que revisa la nómina
contabilizada	varchar(255)		Si	Usuario que contabiliza la nómina
aprobada	varchar(255)		Si	Usuario que aprueba la nómina
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la base de datos
version	numeric(6)		Si	Se utiliza para la concurrencia
otrospagos	numeric(6)		Si	Otros pagos que se le reportan al trabajador
fondo	numeric(8)		Si	Cantidad de tiempo que debe trabajar
idcentrocosto	numeric(19)	FK	Si	identificador del centro de costo
idcuenta	numeric(19)	FK	Si	Identificador de la cuenta
contribucionespec	numeric(8)		Si	Contribución especial

Tabla 48: Descripción de la Tabla dat_registroimpuestotipoajuste

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idregistroimpuestotipoajuste	numeric(19)	PK	No	Identificador del registro de impuesto tipo de ajuste
idtipoajuste	numeric(19)	FK	No	Identificador del tipo de ajuste
idimpuesto	numeric(19)	FK	No	Identificador del impuesto
fechaeliminado	date		Si	Se utiliza para no eliminarlo físicamente de la bases de datos
versión	numeric(6)		No	Se utiliza para la concurrencia

Tabla 49: Descripción de la Tabla dat_comprobanteoperaciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idcomproboperaciones	numeric(19)	PK	No	Identificador del comprobante de operaciones
idnomina	numeric(19)	FK	No	Identificador de la nómina
numero	numeric(4)		Si	Número del comprobante de operaciones
fecha	date		Si	Fecha de realización del comprobante de operaciones
fechaeliminado	date		Si	Se utiliza para no eliminar físicamente el comprobante de operaciones
version	numeric(6)		Si	Utilizado para la concurrencia
descripcion	varchar(255)		Si	Descripción del comprobante de operaciones
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 50: Descripción de la Tabla dat_procesamientopagosadicionales

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idprocesamientopagosadicionales	numeric(19)	PK	No	Identificador del procesamiento del pago adicional
idprocesamientonomina	numeric(19)	FK	No	Identificador del procesamiento de la nómina
idpagosadicionales	numeric(19)	FK	No	Identificador del pago adicional
importe	numeric(8)		No	Cantidad que s ele paga al trabajador por el pago adicional
idtrabajador	numeric(19)	FK	Si	Identificador del trabajador
idelementogasto	numeric(19)	FK	Si	
versión	numeric(6)		No	Se utiliza para la concurrencia

Tabla 51: Descripción de la Tabla dat_procesamientoincidencia

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idprocesamientoincidencia	numeric(19)	PK	No	Identificador del procesamiento de la incidencia
idprocesamiendonomina	numeric(19)	FK	No	Identificador del procesamiento de la nómina
importe	numeric(8)		No	Resultado del procesamiento de la incidencia
versión	numeric(6)		Si	Se utiliza para la concurrencia
idtrabajador	numeric(19)	FK	Si	Identificador del trabajador
idincidencia	numeric(19)	FK	No	Identificador de la incidencia
idelementogasto	numeric(19)	FK	Si	Identificador del elemento de gasto
idcuenta	numeric(19)	FK	Si	Identificador de la cuenta

horas	numeric(8)		Si	Cantidad de horas que tuvo la incidencia
comportamiento	varchar(6)		Si	Comportamiento de la incidencia en cuanto al salario del trabajador, si suma o resta

Tabla 52: Descripción de la Tabla dat_procesamientoretenciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idprocesamientoretenciones	numeric(19)	PK	No	Identificador del procesamiento de la retención
idprocesamientonomina	numeric(19)	FK	No	Identificador del procesamiento de la nómina
idregistroretenciones	numeric(19)	FK	No	identificador del registro de retenciones
importe	numeric(8)		Si	Importe retenido al trabajador
versión	numeric(6)		Si	Se utiliza para la concurrencia
idtrabajador	numeric(19)	FK	Si	Identificador del trabajador
idcuenta	numeric(19)	FK	Si	Identificador de la cuenta

Tabla 53: Descripción de la Tabla nom_tipoajustenomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtipoajuste	numeric(19)	PK	No	Identificador del tipo de ajuste
denom	varchar(255)		Si	Descripción del tipo de ajuste
desde	date		Si	Fecha en la que se inserta el tipo de ajuste
hasta	date		Si	Fecha en la que se elimina el tipo de ajuste
código	numeric(19)		No	Código para el tipo de ajuste

acumulavacdias	numeric(1)		No	Si acumula o no días de vacaciones
acumulavacimporte	numeric(1)		No	Si acumula o no importe de vacaciones
idelementocomun	numeric(19)	FK	Si	Identificador del elemento de gasto
idcuentavacaciones	numeric(19)	FK	SI	identificador de la cuenta

Tabla 54: Descripción de la Tabla nom_tiponomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtiponomina	numeric(19)	PK	No	Identificador del tipo de nómina
denom	varchar(255)		No	Descripción del tipo de nómina
desde	date		No	Fecha en la que se inserta el tipo de nómina
hasta	date		Si	Fecha en que se elimina el tipo de nómina

Tabla 55: Descripción de la Tabla nom_estadonomina

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idestado	numeric(19)	PK	No	Identificador del estado
denom	varchar(255)		No	Descripción del estado
desde	date		No	Fecha inicio del estado
hasta	date		Si	Fecha fin del estado

Anexo # 7: Componente Submayor

Figura 7: Modelo de Datos del componente Submayor

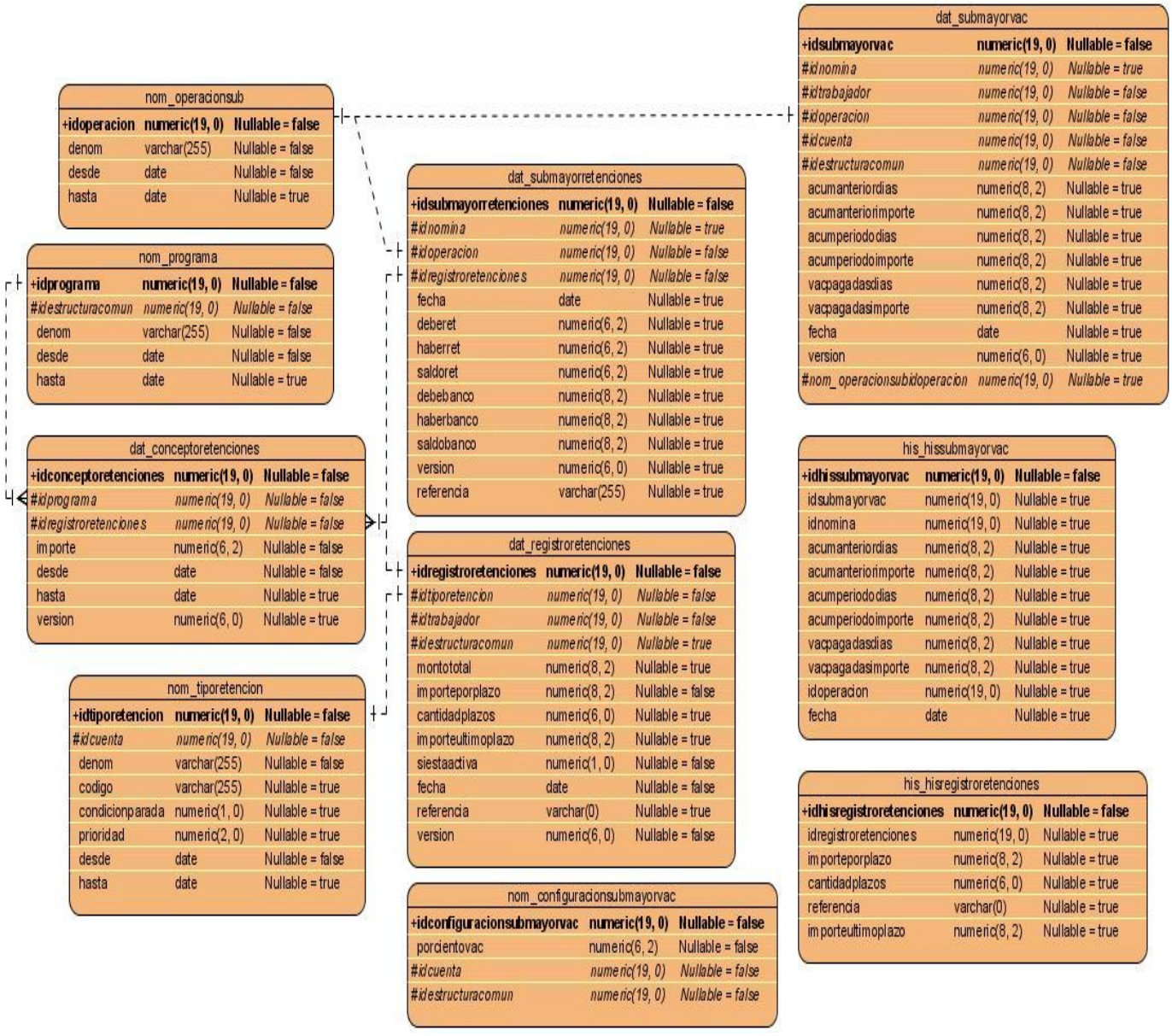


Tabla 56: Descripción de las tablas del componente Submayor

Tablas	Descripción
dat_conceptoretenciones	Tabla que almacena los conceptos de retenciones.
nom_tiporetencion	Almacena los tipos de retenciones existentes.
nom_operacionsub	Almacena los tipos de operaciones que se realizan sobre los submayores.
nom_configuracionsubmayorvac	Se almacena la configuración (la cuenta y el porciento que se aplicara para pagar las vacaciones) del submayor de vacaciones.
his_hissubmayorvac	Almacena un historial del submayor de vacaciones.
his_hisregistroretenciones	Almacena un historial del registro de retenciones.

Tabla 57: Descripción de la Tabla dat_submayorretenciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idsubmayorretenciones	numeric(19)	PK	No	Identificador del submayor de retenciones
idnomina	numeric(19)	FK	Si	Identificador de la nómina
idoperacion	numeric(19)	FK	No	Identificador de la operación
idregistroretenciones	numeric(19)	FK	No	Identificador de registro de retenciones
fecha	date		Si	Fecha del sistema al actualizarse el submayor
deberet	numeric(6)		Si	Cantidad de dinero retenido al trabajador
haberret	numeric(6)		Si	Cantidad de dinero que se le debe retener al trabajador

saldoret	numeric(6)		Si	Diferencia entre el deberet y el haberret
debebanco	numeric(19)		Si	Cantidad de dinero que la empresa le pago al banco
haberbanco	numeric(19)		Si	Cantidad de dinero que la empresa le debe pagar al banco
saldobanco	numeric(19)		Si	Cantidad de dinero que queda por pagarle al banco
version	numeric(6)		No	Se utiliza para la concurrencia
referencia	varchar(255)		Si	Documento que emite la sucursal bancaria que lo aprobó

Tabla 58: Descripción de la tabla dat_submayorvac

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idsubmayorvac	numeric(19)	PK	No	Identificador del submayor de vacaciones
idnomina	numeric(19)	FK	Si	Identificador de la nómina
idtrabajador	numeric(19)	FK	No	Identificador del trabajador
idoperacion	numeric(19)	FK	No	Identificador de la operación
acumanteriordias	numeric(8)		Si	Acumulado anterior en días
acumanteriorimporte	numeric(8)		Si	Acumulado anterior en importe
acumperiododias	numeric(8)		Si	Acumulado periodo días
acumperiodoimporte	numeric(8)		Si	Acumulado periodo importe
vacpagadasdias	numeric(8)		Si	vacaciones pagadas por días
idcuenta	numeric(19)	FK	No	Identificador de la cuenta

vacpagadasimporte	numeric(8)		Si	Vacaciones pagadas por importe
fecha	date		Si	Fecha del sistema
version	numeric(6)		Si	Se utiliza para la concurrencia
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 59: Descripción de la Tabla dat_registroretenciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idregistroretenciones	numeric(19)	PK	No	Identificador del registro de retenciones
idtiporetencion	numeric(19)	FK	No	Identificador del tipo de retención
idtrabajador	numeric(19)	FK	No	Identificador del trabajador
montototal	numeric(8)		Si	Monto total de retenciones
importeporplazo	numeric(8)		Si	Importe de retenciones por plazo
cantidadplazos	numeric(8)		Si	Cantidad de plazos
importeultimoplazo	numeric(8)		Si	Importe del ultimo plazo
siestaactiva	numeric(1)		No	Indica si esta activa o no la retención
fecha	date		No	Fecha del sistema
referencia	varchar(255)		Si	Sucursal bancaria donde se debe pagar la sucursal
version	numeric(6)		No	Se utiliza para la concurrencia
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 60: Descripción de la Tabla dat_conceptoretenciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idconceptoretenciones	numeric(19)	PK	No	Identificador del concepto de retenciones
idprograma	numeric(19)	FK	No	Identificador del programa
idregistroretenciones	numeric(19)	FK	No	Identificador del registro de retenciones
importe	numeric(6)		No	Importe a descontar por la retención
desde	date		No	Fecha en que comienza a estar activo el concepto de retención
hasta	date		Si	Fecha en que deja de estar activo el concepto de retención
version	numeric(6)		No	Se utiliza para la concurrencia

Tabla 61: Descripción de la Tabla nom_programa

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idprograma	numeric(19)	PK	No	Identificador del programa
denom	varchar(255)		No	Descripción del programa
desde	date		No	Fecha en la que se inserta el programa
hasta	date		Si	Fecha en la que se elimina el programa
idestructuracomun	numeric(19)	FK	No	identificador de la entidad

Tabla 62: Descripción de la Tabla nom_tiporetencion

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idtiporetencion	numeric(19)	PK	No	Identificador del tipo de retención
idcuenta	numeric(19)	FK	No	Identificador de la cuenta

denom	varchar(255)		No	Descripción del tipo de retención
código	varchar(255)		Si	Código del tipo de retención
condicionparada	numeric(1)		Si	0- si no tiene, 1- si tiene
prioridad	numeric(2)		Si	Prioridad del tipo de retención
desde	date		No	Fecha en la que se inserta el color de piel
hasta	date		Si	Fecha en la que el color de piel deja de existir

Tabla 63: Descripción de la Tabla nom_operacionsub

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idoperacion	numeric(19)	PK	No	Identificador de la operación
denom	varchar(255)		No	Descripción de la operación
desde	date		No	Fecha en la que se inserta la operación
hasta	date		Si	Fecha en la que se elimina la operación

Tabla 64: Descripción de la tabla nom_configuracionsubmayorvac

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idconfiguracionsubmayorvac	numeric(19)	PK	No	Identificador de configuración del submayor de vacaciones
porcientovac	numeric(6)		No	Porciento de vacaciones
idcuenta	numeric(19)	FK	No	Identificador de la cuenta
idestructuracomun	numeric(19)	FK	No	Identificador de la entidad

Tabla 65: Descripción de la Tabla his_hissubmayorvac

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idhissubmayorvac	numeric(19)	PK	No	Identificador del histórico del submayor de vacaciones
idsubmayorvac	numeric(19)		Si	Identificador del submayor de vacaciones
idnomina	numeric(19)		Si	Identificador de la nómina
acumanteriordias	numeric(8)		Si	Acumulado anterior en días
acumanteriorimporte	numeric(8)		Si	Acumulado anterior importe
acumperiododias	numeric(8)		Si	Acumulado por periodo de días
acumperiodoimporte	numeric(8)		Si	Acumulado por periodo de importe
vacpagadasdias	numeric(8)		Si	Vacaciones pagadas por días
vacpagadasimporte	numeric(8)		Si	Vacaciones pagadas por importe
idoperacion	numeric(19)		Si	Identificador de la operación
fecha	date		Si	Fecha del sistema

Tabla 66: Descripción de la Tabla his_hisregistroretenciones

Atributos	Tipo de dato	PK/FK	Nulo	Documentación
idhisregistroretenciones	numeric(19)	PK	No	Identificador del historial de registro de retenciones
idregistroretenciones	numeric(19)		Si	identificador del registro de retenciones
importeporplazo	numeric(8)		Si	Importe por plazo de la retención
cantidadplazos	numeric(8)		Si	Cantidad de plazos de la retención
referencia	varchar(255)		Si	Referencia de la sucursal bancaria
importeultimoplazo	numeric(8)		Si	Importe del ultimo plazo

GLOSARIO DE TÉRMINOS

Case: Ingeniería se Software Asistida por Ordenador.

Proceso: Conjunto de uno o más procedimientos o actividades directamente ligadas, que colectivamente realizan un objetivo del negocio normalmente dentro del contexto de una estructura organizacional que define roles funcionales y relaciones entre los mismos.

Datos: Hechos conocidos que pueden registrarse y que tienen un significado implícito. Ejemplo: Nombre, Edad, Número telefónico, etc.

Cardinalidad: Número de ocurrencias de una entidad que se relacionan con ocurrencias de la otra entidad.

Herramienta Case: Una herramienta de software que automatiza una parte del ciclo de vida del desarrollo de un software.

Cuello de botella: Parte del sistema que se satura y limita su rendimiento.

Concurrencia: Posibilidad de que varios usuarios modifiquen un mismo registro a la vez.

Entonación de la base de datos: Proceso de realización de ajustes en el contexto o en la BD propiamente dicha para lograr que esta opere con un buen rendimiento.

Índices densos: Todas las claves de búsqueda están en el índice.

Índices no densos: Todas las claves de búsqueda no están en el índice. Solo estarán algunas de ellas. Las que están depende de la arquitectura del índice.

Horas sueldistas: Horas que deben trabajar en un período de pago los trabajadores que tienen un cobro fijo, 192 hrs mensuales y 96 hrs quincenales.

Horas jornaleros: Horas que debe trabajar el trabajador según su sistema de trabajo definido.

Salario escala: Salario básico que cobra el trabajador sin los pagos adicionales ni otros pagos.

Salario devengado: Es el salario que cobra finalmente el trabajador después de haberle sumado los pagos adicionales y descontado las retenciones.