Universidad de las Ciencias Informáticas

Facultad 6

PLUGIN PARA LA MIGRACIÓN DE DATOS DESDE MS ACCESS A POSTGRESQL, PARA LA HERRAMIENTA DE ADMINISTRACIÓN DE BASES DE DATOS HABD

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

AUTOR: David Juan Campos Calás

TUTORES: Ing. Vilmavis La Rosa Sordo

Ing. Flavio E. Roche Rodríguez

Ciudad de la Habana, Cuba

junio 2013

DECLARACIÓN DE AUTORÍA

Por este medio declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) que haga el uso que estime pertinente del mismo.

Para que así conste firmo la presente a los ____ días del mes de junio del 2011.

David Juan Campos Calás
Autor

Ing. Vilmavis La Rosa Sordo
Tutor

Ing. Flavio E. Roche Rodríguez
Tutor

DATOS DE CONTACTO

Tutores:

Ing. Vilmavis La Rosa Sordo

Graduado de Ingeniería Informática en el año 2006 en la Universidad de Ciego de Ávila

Categoría Docente: Instructor

Dirección de la institución: Carretera de San Antonio de los Baños Km 2 ½. Torrens. Ciudad de La Habana, Cuba.

E-mail: vlarosa@uci.cu

Ing. Flavio E. Roche Rodríguez

Graduado de Ingeniería en Ciencias Informáticas en el año 2010 en la Universidad de Ciencias Informáticas

Dirección de la institución: Carretera de San Antonio de los Baños Km 2 ½. Torrens. Ciudad de La Habana, Cuba.

E-mail: feroche@uci.cu

Agradecimientos

No resulta fácil agradecer a todas las personas que han estado conmigo brindándome su apoyo, pues siempre se queda alguien fuera; mas quiero gratificarles de todo corazón por haber estado conmigo, en los momentos buenos y malos de esta parte de mi vida.

Especialmente les quiero agradecer a dos personas, que desde que nací están cargando conmigo. Mi madre Neris de la Caridad Calás Viamonte y Servando Campos Rotger. Para ellos el agradecimiento mayor.

Mi familia ha sido un punto de apoyo clave para poder realizar mis metas. Mi hermanita Dania, con sus frases alentadoras, mi hermana Danay con sus consejos, mis abuelos con sus mimos y mis tíos Juan y Manuel, con sus regalos. A mi querida tía Margot por estar.

Quiero agradecer también a mis tutores Flavio Roche Rodríguez y Vilmavis La Rosa Sordo, mil gracias les doy por su ayuda, en colaboración con ustedes ésta tesis salió adelante.

Quiero agradecer además, a los amigos que me han brindado un rayo de luz en medio de le oscuridad, Manuel Alejandro y su esposa Yaimy, Miosotis, Dariannis: la chiqui, y a Yasmany Uranga.

Dedicatoria

Este trabajo va dedicado a mi pequeña familia. A mi maravillosa esposa Vil, por estar siempre donde más la necesite, y a mi preciosa hija Lia, que con su sonrisa me alumbra el día.

RESUMEN

El software propietario Microsoft Access (MS Access) es uno de los Sistemas Gestores de Bases de Datos (SGBD) que se utilizan en las instituciones y empresas cubanas. Como alternativa en Cuba se potencia el uso de PostgreSQL. HABD, es una herramienta cubana, integral para la administración de bases de datos PostgreSQL, de la que no se pueden beneficiar administradores de BD MS Access. Por el gran volumen de información y las variadas temáticas contenidas en las BD MS Access la opción de obtener una BD PostgreSQL desde cero para almacenarlos y manipularlos, queda desestimada. El objetivo de este trabajo es desarrollar un plugin para HABD, que permita la migración de datos desde MS Access a PostgreSQL. Para la obtención de la solución se analizaron sistemas similares existentes, se planteó un algoritmo para la migración y se emplearon las siguientes herramientas y tecnologías: metodología de desarrollo XP, lenguaje de modelado UML 2.0, Visual Paradigm para UML 8.0 como herramienta de modelado, lenguaje de programación C++, Qt 4.7 como framework y el ambiente integrado de desarrollo Qt Creator 2.2. Entre los aportes se encuentran los criterios teóricos bajo los cuales se realizó la migración, el diseño de un algoritmo que garantiza la migración satisfactoria que puede emplearse, incluso, entre otros gestores con adecuaciones mínimas. El principal resultado es un plugin para HABD que migra datos desde MS Access a PostgreSQL, contribuyendo a la soberanía tecnológica cubana. Las validaciones efectuadas a la solución sustentan resultados acorde a los esperados.

Palabras clave: Base datos, HABD, Migración, MS. Access, PostgreSQL.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo a nivel mundial que alcanza la sociedad está determinado en buena medida por la incidencia de las Tecnologías de la Información y las Comunicaciones (TIC) que en las últimas décadas han acelerado su nivel de renovación. En Cuba, se maximizan los beneficios de las TIC a través de una estrategia de informatización de la sociedad, en la que los organismos, empresas e instituciones nacionales se benefician.

Producto de los procesos empresariales se genera gran cantidad de datos que se conservan para poder hacer análisis que derive información útil. Para una eficiente conservación, uso y fácil acceso a la información, se utilizan, frecuentemente, bases de datos digitales. Se refiere a un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso mediante las TIC. Estas BD funcionan mediante Sistemas Gestores de Bases de Datos (SGBD), que las conservan y permiten su gestión.

En Cuba existe diversidad en cuanto al uso de SGBD. Entre los más comunes se encuentra el software propietario Microsoft Access que integra la suite ofimática de Microsoft (MS Access). Representó en su momento una opción viable por contener una interfaz sencilla e instalarse sin costo adicional respecto al resto de las herramientas de oficina de Windows. Sin embargo el crecimiento de las BD soportadas sobre este gestor ha impactado negativamente en su rendimiento.

El uso de MS Access, es uno de los elementos que ha frenado el proceso de migración hacia software libre y la búsqueda de la soberanía tecnológica del país. Si se tiene en cuenta que el conjunto de datos que se gestiona en las BD de MS Access en Cuba, es tan grande, como variadas son las temáticas que abarcan, queda fundamentado el desecho de la alternativa de rediseñarlas desde cero sobre otro SGBD.

En Cuba se opta por PostgreSQL como alternativa loable a los SGBD propietarios. Este gestor multiplataforma de código abierto, brinda seguridad para integrar los datos y responde eficientemente ante volúmenes de datos significativos.

El Departamento PostgreSQL (DPG), del Centro de Tecnologías de Gestión de Datos (DATEC) de la Universidad de las Ciencias Informáticas (UCI), desarrolla soluciones que facilitan el trabajo con PostgreSQL. Su prioridad es el mejoramiento de las prestaciones de este gestor, contribuyendo así a la soberanía tecnológica cubana. El DPG desarrolla soluciones y brinda asesorías. Entre las soluciones de mayor impacto está la herramienta de administración de BD denominada HABD.

HABD posee una arquitectura orientada a plugin que facilita a los administradores de BD incorporar servicios de administración a PostgreSQL en pocos pasos. Así se le pueden incorporar un gran número de funciones evitando el empleo de varios sistemas para resolver un problema integralmente, pero la BD debe estar ya montada sobre PostgreSQL. Si para las empresas cubanas que utilizan MS Access, PostgreSQL es la alternativa, y como se analizó con anterioridad, el diseño de las BD desde cero es muy costoso llegando a implicar incluso la pérdida de datos históricos, entonces se puede deducir el siguiente problema de investigación: ¿Cómo realizar la migración de datos desde Microsoft Access a PostgreSQL en la herramienta de administración de base de datos HABD?

Para enmarcar la problemática se establece como objeto de estudio: Proceso de migración de bases de datos, delimitado por el campo de acción: Proceso de migración de datos de bases de datos relacionales.

Para poder resolver el problema se establece como objetivo general: Desarrollar un plugin para la herramienta de administración de bases de datos HABD, que permita la migración de datos desde MS. Access a PostgreSQL.

Para dar cumplimiento al objetivo general se proponen los siguientes objetivos específicos:

- Analizar las tendencias y tecnologías referentes a la migración de datos.
- Diseñar e implementar el plugin de migración para la herramienta HABD.
- Validar el sistema propuesto.

Como tareas de la investigación se proponen las siguientes:

- Establecimiento de los conceptos relevantes para el desarrollo del trabajo de acuerdo al punto de vista de varios autores

- Análisis de sistemas que migran base de datos relacionales
- Selección del tipo de migración a utilizar
- Establecimiento de las etapas para efectuar la migración
- Identificación de los requisitos funcionales mediante las Historias de usuario
- Determinación de los requisitos no funcionales que complementarán las Historias de usuario
- Realización del plan de iteraciones
- Obtención de la lista de reserva del producto
- Identificación de las tarjetas Clases-Responsabilidad-Colaboración (CRC)
- Establecimiento de la arquitectura
- Establecimiento de los estándares de codificación
- Implementación las Historias de usuario de acuerdo al plan de iteraciones
- Aplicación de pruebas de funcionalidad
- Corrección de las no conformidades detectadas
- Presentación de los resultados de las pruebas

Para un mejor entendimiento tanto del problema como de la solución, el siguiente trabajo se ha estructurado en tres capítulos, conclusiones parciales en cada uno de ellos y conclusiones generales. A continuación una breve descripción el contenido de los capítulos:

Capítulo 1 Fundamentos teóricos, expone los principales conceptos relacionados con las BD y la migración. Se explican las soluciones existentes y la herramienta HABD. Se estudian y seleccionan las herramientas para modelar, diseñar e implementar el sistema.

Capítulo 2 Características de la solución, describe las características principales que posee el sistema. Se realiza el modelo de dominio, se confeccionan las Historias de usuario y las tareas de ingeniería asociadas a cada Historia de usuario. Para el diseño del sistema se elabora el diagrama de clases que visualiza las tarjetas Clase-Responsabilidad-Colaboración.

Capítulo 3: Implementación y validación, presenta los principales elementos del estándar de codificación y el algoritmo empleado para migrar. Se expone un breve análisis de las formas de

validar las soluciones desarrolladas con XP. Se muestran los resultados de las pruebas realizadas con el objetivo de evidenciar el cumplimiento de las funcionalidades propuestas.

1

CAPÍTULO 1 FUNDAMENTOS TEÓRICOS

Introducción

Para el desarrollo de la investigación se hace necesario establecer los preceptos teóricos a fin de sentar las bases conceptuales que posibilitan una clara comprensión del trabajo. Se presenta la herramienta HABD como solución integral a muchos de los problemas comunes de los administradores de BD. Se analizan características del proceso de migración, sistemas que lo realizan, así como las herramientas y tecnologías a utilizar en el desarrollo de la solución.

1.1 Conceptos asociados al dominio del problema

Una adecuada comprensión del dominio del problema, permite un análisis más certero en busca de la solución, de ahí que se presenten conceptos esenciales.

Datos

Para la informática, los datos son expresiones generales que describen características de las entidades sobre las que operan los algoritmos. Estas expresiones deben presentarse de una cierta manera para que puedan ser tratadas por una computadora. En estos casos, los datos por si solos tampoco constituyen información, sino que ésta se origina del adecuado procesamiento de los datos.[1]

Base de datos

Una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada.[2]

Una base de datos es el lugar donde se guardan los datos en reposo y al cual acceden las diferentes aplicaciones (sistemas o programas) de una organización dada.[3]

Ambos conceptos tributan a que una base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible, en una computadora. Cada registro constituye una unidad autónoma de información. Existen diversas clasificaciones para las BD.

Modelo de Datos

Un modelo, en general, es una representación simplificada de un sistema real. Si la representación es adecuada, podremos interrogar al modelo si deseamos conocer alguna propiedad del sistema real. Un modelo de datos debe poder representar tanto las características estáticas como las dinámicas del sistema real que se pretende modelar. [4]

En consecuencia, un modelo de datos se define por los siguientes componentes[4]:

- Un conjunto de objetos y sus interrelaciones. Esto representa las características estáticas o invariantes e incluye las propiedades de los objetos.
- Un conjunto de operaciones, o lenguaje, que representan las características dinámicas.
- Restricciones sobre los objetos, sus interrelaciones y las operaciones definidas sobre ellos.

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás, que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos permiten modelar la estructura de los datos. Los operadores permiten modelar su comportamiento.[2]

El tipo de BD está muy relacionado con el modelo o paradigma que utiliza. Se clasifican siguiendo variados criterios. Entre los criterios de clasificación, uno de los más utilizados es en función de la tecnología empleada en su funcionamiento. Dentro de dicha clasificación la diversidad radica, fundamentalmente, en la forma de trabajar con los datos y en la concepción o mentalidad que el usuario debe adoptar para interactuar con el sistema.

La investigación estará centrada en las BD de tipo relacional, que son las que comúnmente utilizan los clientes que se benefician con HABD. El modelo relacional propone una base matemática simple y bien definida para el estudio de problemas relacionados con las BD, es menos complejo respecto a otros y ostenta una base conceptual sólida.

Base de Datos relacionales

En las BD relacionales la idea fundamental está dada por el uso de “relaciones”. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados “tuplas”. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).[5]

El lugar y la forma en que se almacenen los datos no tienen relevancia. Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante “consultas” que ofrecen una amplia flexibilidad y poder para administrar la información. Las BD no tienen sentido sin un Sistema Gestor de Base de Datos o Sistemas de Manejo de Bases de Datos como también se le conocen.[5]

Sistema Gestor de Bases de Datos

El software que permite la utilización y/o la actualización de los datos almacenados en una (o varias) base(s) de datos por uno o varios usuarios desde diferentes puntos de vista y a la vez, se denomina sistema de gestión de bases de datos (SGBD)[6]

Los SGDB tienen una serie de funciones esenciales en el trabajo con las BD, dentro de las principales se encuentran [4]:

- Consulta y actualización de los datos: Es la clase más básica de funciones y la única que es visible. Consiste en un conjunto de herramientas que permite a los distintos tipos de usuarios del SGBD extraer, manipular y modificar la información almacenada en la base de datos.
- Mantenimiento de esquemas: El esquema de la base de datos es la descripción de la estructura de la información almacenada en ella. Por ejemplo, para un sistema basado en tablas, el esquema puede consistir en una lista de las tablas en uso, los campos que

contienen, el tipo de datos de cada campo, descripciones en lenguaje natural del propósito de cada tabla y cada campo, y restricciones sobre los valores admisibles en cada campo.

- Manejo de transacciones: Una de las áreas principales de aplicación de los SGBD es lo que se llama procesamiento de transacciones. Una transacción es un programa de aplicación, generalmente de duración breve, que accede y actualiza una parte también generalmente pequeña de la base de datos. Típicos ejemplos son un depósito o extracción de una cuenta bancaria, o una reservación en un vuelo, o una verificación de una tarjeta de crédito.

Existen tantos SGBD como numerosas son sus clasificaciones. Una de las más polémicas es la categorización en cuanto a su distribución. Debido a esta tipificación existen dos grandes grupos: los que constituyen software propietario[1] y los que pertenecen al software libre[2]. Dentro del grupo propietario está Microsoft Access.

MS Access

El SGBD relacional MS Access fue creado por Microsoft para uso personal de pequeñas organizaciones. Permite la creación de BD que pueden ser consultadas por otros programas. Es un sistema interactivo de administración de BD para Windows. Tiene la capacidad de organizar, buscar y presentar la información resultante del manejo de sus BD, como la mayoría de los gestores.

Entre sus principales características se encuentran[7]:

- Es un sistema fácil de utilizar.
- Es sencillo para principiantes y dispone de asistentes mediante los cuales se crean formularios profesionales.

[1]El software propietario, se refiere al tipo de software que tiene limitaciones para ser usado, modificado o redistribuido. Este tipo de software es desarrollado por una empresa o entidad privada y no difunde las especificidades del sistema que comercializa.

2La denominación de software libre abarca programas desarrollados por comunidades de voluntarios. Tiene 4 libertades fundamentales, entre las que se encuentran: libertad de usar el programa con cualquier propósito, de estudiar cómo funciona el programa y modificarlo, adaptándolo a las necesidades de distribuir copias del programa.

- La realización de las réplicas es muy lenta.
- No es multiplataforma, pues solo está disponible para los Sistemas Operativos (SO) Windows.
- No se puede utilizar para proyectos de software que requieran tiempos de respuesta críticos.
- No es un gestor seguro.
- Con grandes volúmenes de datos el rendimiento disminuye y el tiempo de espera de las consultas aumenta.

La estructura de las BD Access se expresa mediante tablas (contenedoras de datos), tipos de datos, llaves foráneas, llaves primarias, y relaciones fundamentalmente. Sin embargo, solo es objetivo migrar los datos, en el presente trabajo.

PostgreSQL

PostgreSQL es un sistema de gestión de BD relacional orientado a objetos y perteneciente al grupo de software libre, publicado bajo la licencia Berkeley Software Distribution (BSD)[1]. Comenzó como un proyecto denominado Ingres en la Universidad Berkeley de California y fue más tarde desarrollado comercialmente por la Relational Technologies/Ingres Corporation. Está ampliamente considerado como el SGBD de código abierto más avanzado del mundo. Posee muchas características, que tradicionalmente sólo se podían ver en productos comerciales de alto nivel, dentro de estas se citan [8]:

- El tamaño máximo de las BD es ilimitado.
- Es un excelente optimizador de consultas.
- Permite el uso de particiones para la mejora de la eficiencia de replicación.
- Muchas de sus versiones admiten la administración de las BD distribuidas.
- Dispone de tipos de datos para aplicaciones específicas
- Cuenta con una comunidad de soporte que da mantenimiento al software de forma directa.

[1]Las licencias BSD pertenecen a la familia de licencias de software libre, van dirigidas principalmente a los sistemas permisibles (Berkeley Software Distribution). Estas tienen menos restricciones en cuanto a la distribución del producto, en comparación con otras licencias de software libre, como permitir el uso del código fuente en software no libre.

- Usa la estrategia de almacenamiento de filas, llamada Control de Concurrencia de Múltiples Versiones (MVCC, por sus siglas en inglés), para conseguir una mejor respuesta ante magnos volúmenes de datos.
- Es multiplataforma y el código fuente es libre.

A pesar de las características desfavorables que hacen de MS Access un gestor poco robusto muchas entidades lo utilizan como su SGBD. Como alternativa latente e ideal, está PostgreSQL que establece un estándar de utilización y calidad en el trabajo con BD. Por la necesidad de independencia tecnológica, en Cuba se está desarrollando el nombrado PostgreSQL Cubano.

PostgreSQL Cubano

El PostgreSQL Cubano, es el SGBD PostgreSQL, al que en su entorno se desarrollan líneas de trabajo para su explotación, entrenamiento y soporte, con el fin de tributar a la soberanía tecnológica de Cuba. La UCI, en conjunto con otras instituciones, potencia el desarrollo y explotación de estas líneas. Va dirigido a resolver las necesidades de las entidades del país. Las tres líneas de trabajo definidas son[9]:

- Desarrollo de materiales para el entrenamiento y soporte, donde se elaborarán materiales e instrumentos para simplificar el trabajo con el gestor.
- Desarrollo de herramientas, donde se desarrollarán aplicaciones para explotar al máximo las potencialidades del gestor.
- Desarrollo del núcleo, donde se añadirán funcionalidades al núcleo para incrementar la potencia y rendimiento del gestor.

Entre las herramientas que brindan valor agregado al PostgreSQL Cubano, se encuentra HABD.

1.2 **Herramienta de administración de bases de datos, HABD**

A HABD se le pueden agregar un gran número de funcionalidades evitando así que los usuarios necesiten de varias aplicaciones para resolver un problema íntegro. Sin contar el paso de avance que significa en materia de soberanía tecnológica.[9]

HABD cuenta con varias funcionalidades basadas en plugin. Actualmente se han desplegado las siguientes:

- Plugin editor de funciones y disparadores.
- Plugin editor de consultas.
- Plugin para la recuperación del estado de entidades.
- Plugin CRUD-PG.
- Plugin diseñador de BD.
- Plugin de generación de datos.
- Plugin de monitorización.
- Plugin para la integración del proceso de normalización de BD relacionales.
- Plugin para el particionado de tablas en BD PostgreSQL.

Como se identificó con anterioridad HABD se basa en el trabajo con BD PostgreSQL y las instituciones cubanas necesitarían llevar las BD MS Access a este gestor para ser administradas. Esta situación se puede resolver a través de la migración de datos.

1.3 **Migración de datos**

La migración de datos es el proceso a través del cual, los datos que se encuentran almacenados, son trasladados hacia otro sitio de destino. Con mucha frecuencia se lleva a cabo la migración de datos debido a la necesidad de las entidades de cambiar el sistema de información usado, en el momento que determinan seleccionar otro sistema gestor de base de datos, o en el tiempo que se modernizan los sistemas y entonces los datos no son compatibles con el sistema nuevo, o cuando se le realiza una mejora a una nueva versión del sistema. Los tipos de migración de datos se pueden clasificar de dos formas atendiendo a su tiempo de uso o ejecución[10]:

- Única migración: Es un proceso de migración de datos que se lleva a cabo una sola vez, sin importar la razón por la cual se esté realizando la migración (cambio de aplicación, cambio de base de datos, cambio de hardware, etc.). Se realiza dentro de la organización, sin incluir las pruebas del proceso que se realizaron, previo a la migración del sistema.

- Migración continua: Procedimientos de migración de datos que se realizan continuamente dentro de la organización. Estas migraciones continuas se pueden presentar a través de interfaces entre sistemas que necesitan comunicar información continuamente entre ellos.

La investigación se centra en la migración única, pues la necesidad impone llevar los datos de MS Access al gestor PostgreSQL. Bastará realizar la migración solo una vez. Con el objetivo de ejecutar correctamente la migración de datos es indispensable emplear técnicas como las que a continuación se mencionan[10]:

- Planeación: Lo más importante al migrar una base de datos es llevar a cabo un proceso de planeación y análisis del trabajo, puesto que aunque pareciera tomarse algún tiempo adicional, éste será retribuido en el éxito de la operación y menos costos por errores de datos.
- Contador de registros: Si la migración se realiza de forma manual, mediante alguna consulta de inserción es recomendable inicializar un contador para cada registro insertado con éxito y otro para los no insertados, así obviamente, la suma de ambos debe ser igual a los registros originales.
- Mapeado de tipos de datos: Algunas plataformas no soportan algunos tipos de datos, así que es necesario planificar el mapeo de los campos en la nueva base de datos.
- Codificación de Caracteres: Cuando el copiado se realiza de forma automática, es necesario identificar la codificación de caracteres que la BD destino espera, pues así se evitará el reemplazo automático de caracteres o en su caso, pérdida de los mismos.

Estas técnicas son de obligatoria utilización en el desarrollo del plugin, pues contribuyen a realizar el proceso migración con calidad. A su vez se necesita de etapas en las que se aplicarán las técnicas presentadas.

Etapas de la migración de datos

La migración de datos comprende varias etapas, que contribuyen a lograr la fiabilidad. Las etapas se presentan a continuación [10]:

- Levantamiento de información: Se realiza un análisis en la fuente de origen, de la información existente definiendo la requerida por el SGBD destino.
- Mapeo de datos: Se mapean los campos entre las fuentes de origen y destino.
- Empleo del ETC (Extractor, Transformador y Cargador de datos): Se utiliza una herramienta ETC, para cargar los datos hacia las tablas de almacenamiento intermedio.
- Análisis de Calidad: Se verifica la calidad de los datos.
- Transformación: Se agregan los campos necesarios en las tablas de almacenamiento intermedio en las que se insertan valores que se obtienen de otras tablas del sistema mediante consultas y funciones.
- Limpieza de datos: Contempla la depuración de los datos en las tablas intermedias.
- Migración de datos: Se realiza la migración de los datos desde las tablas de almacenamiento intermedio hacia las del sistema.
- Pruebas a los datos migrados: Se realizan las pruebas de la migración, determinando que la información sea traspasada con éxito hacia la fuente de destino.

Es necesario destacar que estas etapas se establecen para migrar datos de una organización íntegra, en la que coexisten diferentes orígenes de datos. Fundamentada, entonces, la necesidad de crear tablas de almacenamiento intermedio que posibiliten hacer las adecuaciones necesarias antes de comenzar el proceso migración. En la solución que se presenta, como está bien definida la fuente y se trata de sola una, se puede prescindir de las etapas 3, 4, 5 y 6 asociadas a las tablas de almacenamiento intermedio.

Existen múltiples herramientas que efectúan la migración entre SGBD. De ellas se necesita hacer un análisis enfocado a las principales características, a fin de identificar aspectos reutilizables dentro del proceso de migración, que pudieran aplicarse en el marco que ocupa el presente trabajo.

1.4 **Análisis de las soluciones existentes**

El análisis de las soluciones existentes posibilitará establecer criterios a tener en cuenta para la obtención satisfactoria de la solución. Se tomaron como referencia soluciones que realizan actividades de migración relacionadas con los SGBD que se tratan en el trabajo.

Apatar

Apatar es una herramienta libre, líder en la integración de datos. Realiza una amplia gama de ETC (Extracción, Transformación, and Carga)[1] y otras funcionalidades útiles a la hora de migrar datos. Permite reducir la integración de costos, transfiere los datos correctamente sin errores, acorta el tiempo de implementación, elimina la costumbre de codificación gracias a su diseño visual. No se requiere codificación para lograr incluso una compleja integración. Cuenta con una ayuda en línea y una comunidad de soporte. Apatar es una herramienta bajo la licencia General Public License (GPL)[2]y la licencia comercial perteneciente al grupo de software de código abierto.[11]
La principal desventaja de este sistema es que la migración se hace lenta y tediosa al tener que implementar tabla por tabla.

Kettle

Kettle es una es una herramienta capaz de manipular y transformar información en proyectos en los que se necesita trabajar con datos. Su solución para cubrir las necesidades de extracción, manipulación, validación y carga de datos desde múltiples fuentes de origen y en diferentes entornos puede resolver diversas problemáticas.[7]

[1]ETC: También como ETL por sus siglas en inglés, es el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos y limpiarlos, y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

[2]La Licencia Pública General de GNU (GNU GPL) garantiza a los usuarios finales (personas, organizaciones, compañías) la libertad de usar, estudiar, compartir (copiar) y modificar el software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios

Esta herramienta es fácil de utilizar y permite hacer migraciones de forma rápida. Ofrece una serie de ventajas de las que se pueden citar[7]:

- Exportar BD a ficheros u otras BD.
- Importar BD a ficheros en formato Excel o texto.
- Migración de datos entre diferentes BD.
- Explotación de los datos existentes en BD (tablas, vistas, sinónimos).
- Enriquecer la información mediante búsqueda de datos en diferentes almacenes de información (BD, ficheros de texto, hojas Excel).
- Limpieza de datos aplicando transformaciones de datos con condiciones complejas.

En el marco de la necesidad de migrar datos que se presenta, de Kettle se puede eliminar las tablas intermedias, pues no se necesitan realizar el proceso de migración desde múltiples fuentes de datos.

SwisSQL

SwisSQL es una herramienta de migración de datos propietaria muy abarcadora, que ayuda a la migración y la transferencia de esquemas de BD y datos a través de BD muy empleadas como Oracle, IBM DB2, MS-SQL Server, Sybase, MySQL, PostgreSQL y MS Access. Ofrece un proceso de migración continua, fácil de usar y extensible, garantiza la integridad y fiabilidad de los datos.

Características esenciales[11]:

- Admite la migración y/o transferencia de esquemas y datos a través de Oracle, IBM DB2, SQL Server, Sybase, MySQL, MaxDB, MySQL, PostgreSQL y BDMS Access.
- Migra tablas, índices, restricciones, junto con los datos.
- Es multiplataforma

Entre las principales ventajas que brindan se encuentran [11]:

- Migración flexible, abierta y extensible proceso.
- Migración rápida a través de BD asegurando la integridad de datos sin pérdida de información.
- Migrar/transferir datos entre BD desplegadas en diferentes plataformas (Windows, Unix o GNU/Linux).

- Proporciona un enfoque uniforme para la migración a través de diferentes BD.
- Ofrece un mecanismo único para verificar los datos migrados, mediante el uso del “Verificador de migración de BD”.

Aunque realiza la migración de datos de conjunto con la estructura, está pensada para procesos de migración continua que difiere de los propósitos del caso que ocupa

Kexi

Kexi organiza la información desde una hoja de cálculo. Establece consultas a servidores de BD entre los que se encuentran MySQL y PostgreSQL.

En muchos aspectos el modelo de trabajo es similar al de MS Access, por ejemplo en los macros. Tiene plena compatibilidad con lenguajes como Python y Ruby para programar scripts. Éstas se realizan fácilmente debido a que las tablas y formularios de las BD se guardan en un único archivo. Otra característica más de la versatilidad de Kexi la encontramos en que funciona en los tres sistemas operativos principales: Windows, Macintosh y GNU/Linux. En resumen, es una herramienta para el manejo de datos muy completa y con grandes posibilidades de interacción con otros programas y tecnologías.[11]

La esencia de Kexi es el manejo de datos similar al que hace Access. Exporta datos hacia varios gestores a los que les puede realizar consultas, lo que difiere en esencia con este trabajo.

Database Migration Toolkit Professional

Database Migration Toolkit Professional es un programa que permite convertir entre formatos de BD diferentes. Ofrece un sencillo asistente paso a paso, que permite conectarse a la base de datos de origen, seleccionar las tablas y luego convertir y copiar al destino. Permite optar por incluir todos los datos o simplemente copiar la estructura de la base de datos, brinda todas las opciones de conversión necesarias, teniendo en cuenta las peculiaridades de los dos formatos de entrada y salida de base de datos. Esta aplicación tiene entre sus características las siguientes[11]:

- Se destaca por ser el más intuitivo y una herramienta completa en su categoría.

- Reduce el esfuerzo, coste y riesgo de migración.
- Soporta una serie de Sistemas de Base de Datos (SBD) como son Oracle, MySQL, SQL Server, PostgreSQL, IBM DB2, SQLite, FireBird, MS Access, Paradox y Lotus.

Esta herramienta propietaria se basa en la carga de la base de datos origen, presentándola en tablas y permite la selección de aquellas tablas que desea migrar, brindando la posibilidad de escoger la estructura solamente. Esta forma de trabajo se considera acertada en trabajos de migración de datos.

Talend Open Studio

Talend Open Studio es una herramienta basada en código abierto de integración de datos. Ofrece soluciones de calidad de datos, totalmente complementarias con sus soluciones de integración de datos. Se emplea para ETC, sincronización o replicación de BD, migración y transformación: Incluye incluso una vista gráfica de procesos que aplica a BD

Entre sus principales características se encuentran[11]:

- Plataformas: Se ejecuta en Windows, Unix y GNU/GNU/Linux.
- Código fuente: El código fuente Java/Eclipse está disponible para su descarga y personalización.
- Soporte: Wiki y foro de Talend.
- BD: MySQL, MS SQL Server, DB2, Oracle, Ingres, PostgreSQL, Sybase, MS Access, Informix, Firebird.

Como puede verse abarca mucho más que la migración de datos que es lo que se pretende hacer en HABD. Está pensada como suite de integración de datos y los requerimientos de hardware no son bajos por todas las funcionalidades que contiene.

Del análisis de los sistemas anteriores, que permiten migrar datos, cada uno con sus peculiaridades, se determinó que la propuesta de este trabajo debe permitir listar las tablas del gestor de origen para que el usuario seleccione las que desea llevar a PostgreSQL. Se debe, además, garantizar la

coexistencia de los datos que se migren con los que ya tenga la BD destino. La conexión debe basarse en protocolo de acceso a datos ODBC que es el que utilizan la mayoría de los gestores.

1.5 **Herramientas y tecnologías a utilizar**

A partir de la comprensión de los conceptos, el análisis del objeto de estudio y el campo de acción, es necesario presentar las herramientas y tecnologías con que se desarrollará la solución. Debe aclararse que en principio deben ser las establecidas para HABD a fin de respetar su arquitectura. Esto contribuye a la estandarización de las nuevas funcionalidades que se le incorporen.

Metodología de desarrollo

La metodología de desarrollo de software en Ingeniería de Software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Es un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. [12]

El uso de metodologías, guía a los desarrolladores de software durante todo el ciclo de vida, sin embargo para la acertada selección se deben tener en cuenta las características del software a obtener, el entorno para el que se desarrolla y el equipo de trabajo.

Metodología Extreme Programming

La metodología ágil XP (eXtreme Programming en español Programación Extrema) es un conjunto de valores, principios y prácticas para el desarrollo vertiginoso de un software de alta calidad que proporciona el valor más alto para el cliente en el menor tiempo posible. Fue creada por Kent Beck, en esta se utiliza un enfoque orientado a objetos y con su empleo se logra la confección de productos fáciles de usar, rápidos, fiables y robustos contra los fallos.[13]

XP se centra en la satisfacción del cliente, que se considera miembro del equipo. Potenciar al máximo el trabajo en equipo y enfatizar la convivencia con el cambio, son características que hacen que esta metodología, se ajuste a proyectos con requisitos imprecisos, muy cambiantes y donde

existe un alto riesgo técnico y el objetivo sea obtener una versión funcional en la menor brevedad de tiempo posible.

Esta metodología ofrece muchas ventajas para el proceso de desarrollo de software. En un proyecto relativamente corto se hace más eficaz el trabajo. Se ahorra tiempo en documentación y los miembros del equipo se enfocan en poner todo sus esfuerzos en la realización del producto, que resultan más fiables y robustos contra fallos debido al diseño de las pruebas de forma previa a la codificación. Posibilita, además, la existencia de pocos errores y los requisitos obtenidos pueden ser modificados con gran facilidad. Cualquier miembro del proyecto puede desempeñar el rol de desarrollador, quedando la posibilidad de que el código siempre se mejore y se simplifique usando sistemas para la gestión de las versiones para evitar la duplicación de trabajo.

La Metodología XP, se adapta al desarrollo del Plugin, que es un proyecto pequeño sin muchas funcionalidades. Además necesita de una versión funcional en poco tiempo.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.[14].

Mediante UML se describirá el sistema que se propone en términos de la Vista lógica y los diagramas de clases fundamentalmente.

Herramienta para el modelado

Las herramientas para el modelado, también llamadas CASE (por Ingeniería de software asistida por computadora en inglés) son un conjunto de métodos, utilidades y técnicas que dan asistencia a los analistas, ingenieros de software y desarrolladores, destinadas a facilitar el desarrollo de software incrementando su productividad y disminuyendo el costo de tiempo y de dinero.

Visual Paradigm para UML 8.0

Visual Paradigm para UML es una herramienta CASE multiplataforma de modelado UML, que soporta el ciclo de vida completo del desarrollo de software. Permite dibujar todos los tipos de diagramas de clases, genera código desde diagramas, realiza ingeniería inversa e informes en varios formatos Esta herramienta presenta licencia gratuita y comercial.

Entre las ventajas por las que sobresale se encuentran[15]:

- Navegación intuitiva entre código y el modelo.
- Poderoso generador de documentación y reportes UML PDF/HTML/MS Word.
- Demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente.
- Superior entorno de modelado visual.
- Soporte completo de notaciones UML.
- Diagramas de diseño automático sofisticado.
- Análisis de texto y soporte de tarjeta CRC.
- Proporciona soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas java.

Visual Paradigm para UML 8.0 es la herramienta de modelado a utilizar, pues soporta todo el ciclo de vida de desarrollo de un software, además de garantizar la producción de código para la mayoría de los lenguajes de programación como C++, en el cual está implementada HABD.

Lenguaje de Programación

Un lenguaje de programación puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen la estructura y el significado de sus elementos, respectivamente.[16]

Lenguaje de Programación C++

C++ es un lenguaje orientado a objetos derivado del C. Es de propósito general y se le han añadido nuevos tipos de datos, clases, plantillas, mecanismo de excepciones, sistema de espacios de nombres, funciones en línea, sobrecarga de operadores, referencias, operadores para manejo de memoria persistente, y algunas utilidades adicionales de librería.

Con C++ se puede implementar desde sistemas operativos y compiladores hasta aplicaciones de BD y procesadores de texto, pasando por juegos y aplicaciones. Una de sus propiedades es la reutilización del código en forma de librerías de usuario. Proporciona un acceso a bajo nivel de hardware solamente igualado por el ensamblador. El C++ mantiene las ventajas de C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Ha eliminado algunas de las dificultades y limitaciones de C.

Para el desarrollo del plugin se resuelve utilizar C++ como lenguaje de programación, por las características que lo definen como lenguaje versátil. Posibilita una integración apropiada con PostgreSQL pues sus librerías fueron programadas en los lenguajes C y C++. Además, el plugin a desarrollar se integrará a la herramienta HABD, que se basa en este lenguaje.

Framework de desarrollo

Un framework, en el desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Framework de desarrollo Qt 4.7

QT 4.7 es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores. Utiliza el lenguaje de programación C++ de forma nativa, adicionalmente puede ser utilizado en varios lenguajes de programación. Funciona en las principales plataformas, y tiene un amplio apoyo. La

interfaz de programación de aplicaciones (API) de la biblioteca cuenta con métodos para acceder a BD mediante SQL, así como uso de XML, gestión de hilos, soporte de red, una API multiplataforma unificada para la manipulación de archivos y una multitud de otros para el manejo de ficheros, además de estructuras de datos tradicionales. Esta biblioteca se encuentra distribuida bajo los términos de GNU Lesser General Public License, Qt es software libre y de código abierto.[17]

Se emplea el framework en el desarrollo para el plugin de migración, debido a las siguientes características [9]:

- Rápido para el desarrollo de aplicaciones.
- La utilización de C++ de forma nativa le hace menos vulnerable a los fallos y responde con una rápida velocidad por estar escrito en dicho lenguaje.
- Provee varias clases para facilitar ciertas tareas de programación como la comunicación con BD, pues entre sus módulos se encuentra QtSQL, que permite la interacción con BD.
- Presenta un buen diseño orientado a objetos y una mayor reutilización de código para agilizar el proceso de desarrollo de aplicaciones visuales, lo que trae consigo que sea más fácil el trabajo diario del programador.

Entorno integrado de desarrollo

Un Entorno Integrado de Desarrollo, IDE por siglas en inglés, es una aplicación visual que sirve para la construcción de aplicaciones a partir de componentes.[18] Se componen por un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser software independiente o formar parte de otras aplicaciones.

Entorno integrado de desarrollo: QtCreator 2.2

El entorno de desarrollo integrado Qt Creator posibilita el desarrollo aplicaciones en C++ de manera sencilla y rápida. Se basa en la librería Qt y posee herramientas para la administración y construcción de proyectos, depurador visual, GUI (Interfaces gráficas de usuario) integrada, diseñador de formularios y auto-completado de código. Este IDE está disponible para las plataformas: GNU/Linux, Mac OSX; Windows, Symbian y Maemo.[19]

1.6 **Conclusiones Parciales**

En este capítulo se sentaron las bases teóricas necesarias para poder comprender el proceso de migración de BD lo que posibilitó determinar que la propuesta se basa en el proceso de migración única de datos provenientes de BD relacionales de MS Access a PostgreSQL. Como contribución a la soberanía tecnológica y para ampliar las facilidades y funcionalidades que brinda HABD a los administradores de BD se puede concluir que es necesario que la solución sea un plugin para esta herramienta. Del estudio de sistemas que realizan el proceso de migración de datos se tomaron elementos para el desarrollo del plugin como la selección de las tablas que contienen los datos a migrar, la permisión de la coexistencia entre los datos de origen y los que estén en el destino, y el uso del protocolo ODBC, esencialmente. Por otro lado se caracterizaron las herramientas y tecnologías a utilizar, según la arquitectura especificada para HABD, prefijando como metodología de desarrollo XP, lenguaje de modelado UML 2.0 y Visual Paradigm para UML 8.0 como herramienta CASE, para la implementación del plugin C++ como lenguaje de programación, el framework Qt 4.7 y QtCreator 2.2 como IDE de desarrollo.

2

CAPÍTULO 2CARACTERISTICAS DEL PLUGIN

Introducción del capítulo

Luego de haber identificado una solución a la necesidad evidenciada en la problemática y hacer un estudio del arte, se propone el siguiente capítulo en el que se describirán las características principales que posee el sistema. Se realiza el modelo de dominio, se confeccionan las Historias de usuario y las tareas de ingeniería asociadas a cada una de ellas. Para el diseño del sistema se elabora el diagrama de clases que visualiza las tarjetas Clase-Responsabilidad-Colaboración. De esta forma se tendrá un entendimiento más amplio de la propuesta del plugin.

2.1 Identificación de la solución

El DPG, perteneciente al centro DATEC, desarrolló la herramienta de administración de BD HABD, cuya principal finalidad es la administración del PostgreSQL Cubano aunque también puede utilizarse en otros contextos. HABD con una arquitectura basada en plugin, carece de funcionalidades que permitan la transformación de los datos de BD en MS Access a PostgreSQL

La esencia de este trabajo es obtener un plugin que permita la migración de datos desde Access a PostgreSQL que sea perfectamente acoplable a HABD, del que se puedan escoger las tablas cuyos datos se desean, que se visualicen previamente y que una vez migrados no entre en conflictos con los ya existentes en la BD destino. Los aportes prácticos serían de especial importancia para las entidades y otros usuarios que necesiten migrar los datos de sus BD establecidas en el gestor de Microsoft, al poder llevarlas un gestor más robusto sin perder información y, al mismo tiempo, adquirir un producto nacional en pos de la soberanía tecnológica.

2.2 Modelo conceptual

El modelo conceptual resulta una herramienta de comunicación fundamental que obliga y permite a desarrolladores a pensar formalmente en el problema y a validar su comprensión del mismo.

Mediante este modelo se establece además un vocabulario propio del problema; y junto a los requerimientos constituye la entrada más importante para el diseño. Es importante aclarar que por lo general se requieren varias iteraciones para obtener un buen modelo. [20]

El modelo conceptual, como el modelo de dominio es una representación visual estática que se realiza con el objetivo de lograr mejor comprensión del contexto para la obtención de requisitos del sistema. Básicamente, de manera visual no es más que un diagrama de clase UML que modela los conceptos básicos asociados al dominio del problema, sus propiedades más importantes y las relaciones que resultan imprescindibles para contextualizar dichos conceptos, a fin de poder brindar elementos que permitan la identificación de los requisitos del sistema. Además es válido para cualquier metodología de desarrollo de software.

Modelo Conceptual

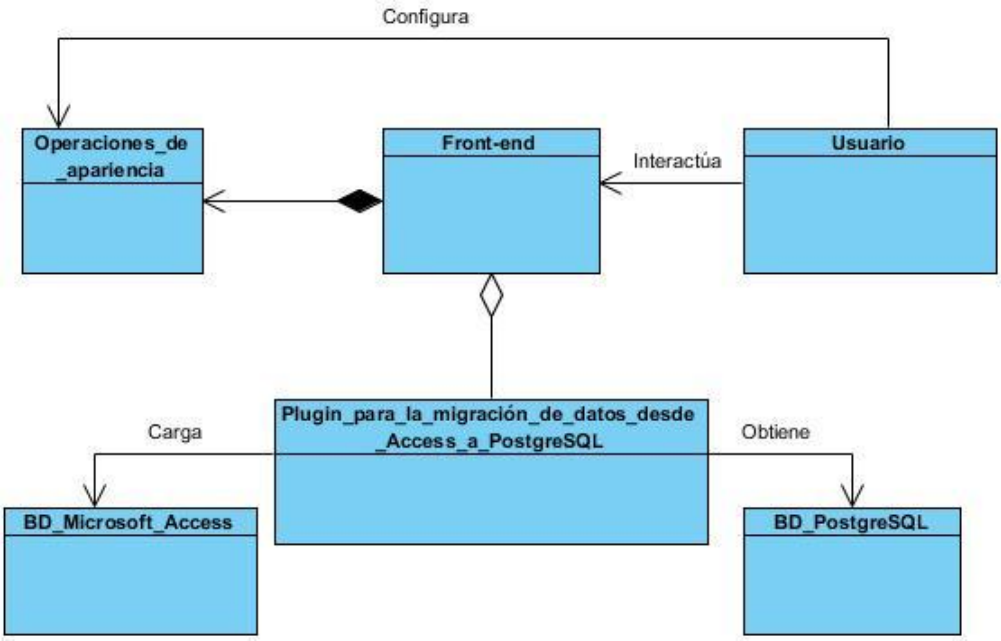

Figura 1 Modelo Conceptual

- Usuario: Persona que interactúa con el front–end y configura las opciones de apariencia incorpora/quita plugin)

- Front-end: HABD a la cual se le incorporan los plugin por su arquitectura y para su funcionamiento, en este caso el de migración de BD.
- Operaciones_de_apariencia: Opciones que le permiten al usuario realizar cambios en la apariencia del Front-End.
- Plugin_para_la_migración_de_datos_desde_Access_a_postgreSQL: Plugin que le permite al usuario migrar datos de MS Access a PostgreSQL
- BD_Microsoft_Access: BD origen a la que el plugin se conecta y extrae los datos.
- BD_PostgreSQL: BD destino hacia donde se migran los datos extraídos de la BD MS Access.

2.3 **Propuesta del sistema**

Se propone un plugin para la migración de datos desde MS Access a PostgreSQL, capaz de integrarse a HABD, que permite seleccionar la BD a la que se le migrarán datos desde cualquier localización de la computadora a la que tenga acceso el usuario y elegir la BD para salvar los datos transformados a PostgreSQL. Permitirá que se seleccionen las tablas portadoras de los datos a migrar y que se visualice su estructura.

Luego de planteada la propuesta es necesario que se establezcan los elementos que permitirán la migración de los datos tratando de establecer la mayor fidelidad posible mediante el seguimiento de las etapas de la migración.

2.4 **Migración vista desde la propuesta**

Levantamiento de información

Esta etapa plantea que debe ser bien detallada cuando se realiza la migración de un sistema de información íntegro y cuando las fuentes de datos son diversas. El caso que se presenta solo se establece que se migrarán los datos y la esencia de la estructura que los soporta, además pueden ser seleccionados por el usuario, pero siempre desde MS Access y hacia PostgreSQL. Inicialmente se verificará si la BD destino posee elementos comunes con la BD origen, entiéndase que no es vacía sino que contiene tablas y datos. A partir de ahí se establecerá el proceso de migración, garantizando que se conserven los datos del destino que no tienen puntos coincidentes con los del origen que se desea migrar.

Ejemplo:

Si la BD Origen contiene las tablas TUsuario y TRoles y la BD destino contiene TPersonas y se desea migrar la tabla TUsuario, el resultado será que la BD destino mostrará los registros TUsuario y TPersonas.

Mapeo de datos

El mapeo de datos se realizó a través de la biblioteca QtSql.dll, que es capaz de reconocer gran parte de los tipos de datos de Microsoft Access, tal como se muestra en la siguiente tabla:

Mapeo de datos

Tipos de datos Access	Tipos de datos entendibles por biblioteca QtSql.dll	Tipos de datos convertidos para PostgreSQL (valores)
Sí / No	Int	Numeric (1 / 0)
Text	Qstring	Text (“ ”)
Número	Int	Numeric (1, 2, 3…∞)
Moneda	Double	Float (x.y z)
Fecha/Hora	DateTime	Date (00.00.00)
Auto-numérico	Int	Numeric (1, 2, 3…∞)
Memo	Qstring	Text (“ ”)

Tabla 1.Mapeo de Datos

Se propone además que se respeten las llaves primarias de cada tabla que se migre.

La etapa Empleo de ETC no es necesaria debido a que se prescindirá la tabla intermedia, pues con estructuras de datos para el almacenamiento en tiempo de ejecución es suficiente. La etapa Análisis de Calidad se evidenciará con los resultados en las pruebas del sistema.

Transformación

En esta etapa se obtienen las tablas del sistema mediante consultas y, la utilización de clases con sus funciones y propiedades, de la biblioteca QtSql.dll. Prescindiendo de las tablas intermedias de creación, no se realiza la Limpieza de datos, que contempla su depuración, en este tipo de tablas.

Migración de datos

Se realiza la migración de los datos desde las tablas de la BD origen hacia el SGBD destino y se realizan las pruebas de la migración, determinando que la información sea traspasada con éxito como parte de las pruebas. Para esta etapa se empleará también la biblioteca de QtSql.dll especializada en BD.

Clases Utilizadas

- QSqlDriver: Permite tener acceso a una base de datos SQL específica. Se utiliza para acceder a los datos de la tabla.
- record (): Función que pertenece a la clase QSqlDriver, se utiliza para obtener los campos de una tabla, establecida previamente, en la BD origen.
- QSqlDatabase: La clase representa la conexión a una BD determinada.
- database (): Se utiliza para devolver la conexión tanto de la BD origen como la de destino.
- setDatabaseName (): Es utilizado para nombrar a las BD.
- databaseName (): Se emplea para obtener el nombre de la BD.
- QSqlQuery: Proporciona métodos para ejecutar y manipular sentencias SQL.
- exec (): A través de ésta función se ejecutan las consultas SQL utilizadas para: seleccionar los datos de una tabla (“SELECT tabla FROM (campo)”), crear la tabla a migrar, en la BD destino (“CREATE TABLE nombre_tabla (campo), tipo_campo”) e insertar los datos en las nuevas tablas creadas (“INSERT INTO tabla (campo) VALUES (valor)”).
- isSelect (): Permite conocer si la consulta realizada resultó satisfactoria.
- QSqlField: Se usó para la manipulación de los campos de las tablas.
- name(): Se utilizó para obtener el nombre del campo.
- type (): Se utilizó para obtener el tipo del campo.

Para la descripción de las funcionalidades del sistema, necesarias para satisfacer los pasos anteriores, se presentan las Historias de usuario.

2.5 **Historias de usuario**

Las Historias de usuario son una manera simple de escribir una tarea concisa que aporta valor al usuario o al negocio. Es importante aclarar que no se detalla más allá del momento en que la historia de usuario se vaya a desarrollar. Las mismas constan de tres partes, dos de ellas fundamentales: confección de la tarjeta y confirmación. La primera etapa hace referencia a la descripción escrita en lenguaje de negocio que sirve como identificación y recordatorio de requerimiento y ayuda para la planificación mediante la priorización. La otra etapa y última consiste en pruebas que permitirán decir que la HU se completó con éxito. [21]

Para el presente trabajo de diploma se obtuvo un total de 3 HU. A continuación se muestra la HU de mayor trascendencia “Migrar datos”, el resto se encuentra en el artefacto “Historias de Usuario” presente en el Expediente de Proyecto.

HU Migrar datos

HISTORIA DE USUARI0	
Número: 2	Nombre de la Historia de Usuario: Migrar datos
Cantidad de modificaciones a la Historia de Usuario: ninguna	
Usuario: David Campos Calás	Iteración asignada: iteración 2
Prioridad en negocio: muy alta	Puntos estimados:3 semanas
Riesgo en desarrollo: muy alta	Puntos reales: 3 semanas
Descripción: Se transforman los datos de la BD MS Access previamente cargada, a datos para una BD PostgreSQL asignada a como BD destino. La operación debe realizarse al dar clic en el botón Migrar si se completaron los campos necesarios	
Observaciones: El botón Migrar no se activará hasta tanto no se hayan completado los campos obligatorios relacionados con las BD origen y destino.	

Prototipo de interfaz:

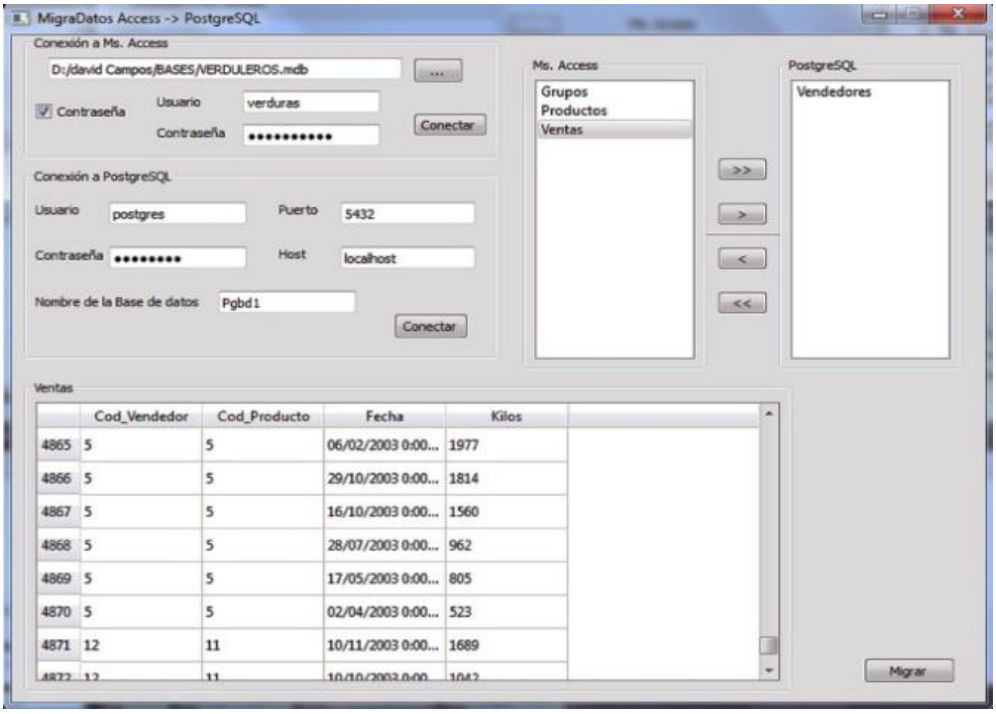

Tabla 2. Historia de usuario Realizar la migración

2.6 **Lista de reserva del producto**

La metodología XP tiene como una de sus actividades fundamentales la confección de la Lista de Reserva del Producto (LRP). En este listado se recoge una lista priorizada de todas las funciones y cualidades que debe realizar el proyecto, esta puede crecer y modificarse a medida que se obtiene mayor conocimiento del producto y las necesidades del cliente. A través de este artefacto debe propiciar que el producto cumpla con los requerimientos técnicos y del negocio.

A continuación se muestra la LRP de la solución:

Lista de reserva del Producto

REQUISITOS FUNCIONALES			
ítem	Descripción	Estimación (semanas)	Estimado por
Prioridad: Muy Alta			
1	Realizar la migración a la base de datos MS Access seleccionada.	3	Programador
2	Conectar la base de datos MS Access cargada	1	Programador
3	Conectar la base de datos PostgreSQL destino	1	Programador
Prioridad: Alta			
4	Cargar la base de datos MS Access que se migrará a PostgreSQL.	0.3	Programador
Prioridad: Media			
	No hay elementos		
Prioridad: Baja			
5	Mostrar las tablas contenidas en la BD origen para seleccionar las que se migrarán	0.3	Programador
6	Visualizar la estructura de una tabla seleccionada	0.3	Programador
Requisitos No Funcionales			
1	Apariencia o interfaz externa: Interfaz organizada y fácil de manejar. Comprensible para usuarios con conocimientos medios en Informática.		Programador

2	Software: El sistema debe ser multiplataforma. Necesita tener instalado los controladores ODBC para la conexión con las BD. Debe estar instalado PostgreSQL 9.1 o superior	Programador
3	Portabilidad: El plugin debe instalarse en HABD.	Programador
5	Restricciones del diseño y la implementación: Como metodología de desarrollo se empleará XP, para el modelado Visual Paradigm para UML 8.0, el lenguaje de programación debe ser C++ y como IDE de desarrollo Qt Creator en la versión 2.2, con framework de desarrollo Qt versión 4.7. Debe ser capaz de acoplarse a HABD. Deben ser instalados los driver ODBC para PostgreSQL y para MS. Access. Debe migrar los archivos de MS Access .mdb y .accdb.	Programador

Tabla 3. Lista de reserva del producto

2.7 **Tareas de ingeniería**

Las Tareas de Ingeniería están fuertemente asociadas a las funcionalidades y características que se deben cumplir. Pertenecen a una HU en específico. Se consideran como la entrada de trabajo para el equipo de programadores. Es una ficha que contiene el número identificador de la tarea, el identificador de la HU con la que está relacionada, el nombre de la tarea, la fecha de inicio, la fecha de fin, el equipo responsable y la descripción.

A continuación se muestra un ejemplo de las tareas de la ingeniería asociadas a las HU. Pueden localizarse todas en el documento Tareas de Ingeniería del proyecto.

Tarea de ingeniería Cargar Archivo MS Access

Tarea de Ingeniería	
Número Tarea: 1	Número Historia de Usuario: 1
Nombre Tarea: Cargar el archivo MS Access.	
Tipo de Tarea : Desarrollo	Puntos Estimados: 0,3
Fecha Inicio: 17/02/2013	Fecha Fin: 20/02/2013
Programador Responsable: David Juan Campos Calás	
Descripción: Al dar clic en el botón que permite localizar la BD MS Access el sistema muestra una interfaz que permitirá identificar la ruta donde se encuentra el archivo que contiene la BD origen, luego de identificada, muestra esta ruta en un cuadro de texto	

Tabla 4. Tarea de ingeniería Cargar Archivo MS Access

2.8 **Plan de iteraciones**

La metodología XP aporta mayor valor a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. “El plan de iteraciones está compuesto por iteraciones que no superan las tres semanas. Este posee una arquitectura del sistema que puede ser establecida en la primera iteración y utilizada durante la trayectoria del proyecto. En la elaboración del plan se deben tomar en cuenta elementos tales como: velocidad del proyecto y tareas no terminadas de iteración anterior.”[22]

Plan de iteraciones

Release	Descripción de la iteración	Orden de la HU a implementar	Duración total (semanas)
1	El objetivo de esta iteración es conectarse las BD tanto origen como destino para lo cual se realizarán las HU correspondientes a conectar a la base de datos PostgreSQL destino de la migración, seleccionar la base de datos MS Access origen, cargar la base de datos MS Access seleccionada.	1. Item 4 de la LRP 2. Item 2 y 3 de la LRP	3
2	El objetivo de esta iteración es que el plugin realice la migración. Es por esto que la iteración va a centrarse en las HU relacionadas con la migración, seleccionar las tablas a migrar, visualizar tablas a migrar y migrar datos.	1. Item 1 de la LRP 3. Item 6 de la LRP 2.Item 5 de la LRP	3

Tabla 5. Plan de iteraciones

2.9 **Arquitectura y Diseño**

Pressman asegura que el diseño es el lugar donde se fomentará la calidad del software. La metodología XP hace especial énfasis en los diseños simples y claros, pues un diseño simple se implementa más rápidamente que uno complejo. El objetivo principal del diseño es especificar una solución que pueda ser fácilmente convertida a código fuente y construir una arquitectura fuerte, simple y extensible. Las tarjetas Clase-Responsabilidades-Colaboración son el diseño las soluciones de XP

Estilo arquitectónico

Un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema que tiene como objetivo establecer una estructura para todos los componentes del mismo. Describe un conjunto de conectores y restricciones que integran dichos componentes. La arquitectura de software define un conjunto de estilos arquitectónicos para sintetizar estructuras de soluciones y son fundamentales para la toma de decisiones del diseño.[23] El seleccionado fue “llamada y retorno”.

El estilo llamada y retorno refleja la estructura del lenguaje de programación. Permite al diseñador de software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Se basa en la abstracción de funciones y persigue ser escalable y modificable. Se caracteriza por tres principales sub-estilos[24]:

- Arquitectura de programa principal-sub rutinas: descompone las funciones en una jerarquía de control donde un programa principal llama a un número de componentes del programa, los cuales pueden también llamar a otros componentes.
- Arquitecturas orientadas a objetos: los componentes de un sistema encapsulan los datos y las operaciones que se deben realizar para manipular a información. La comunicación y la coordinación entre componentes se consiguen a través del paso de mensaje.
- Arquitecturas Estratificadas: se crean diferentes capas y cada una realiza operaciones que progresivamente se aproximan más al cuadro de instrucciones de la máquina. En la capa externa, los componentes sirven a las operaciones de interfaz de usuario. En la capa interna, los componentes realizan operaciones de interfaz del sistema. Las capas intermedias proporcionan servicios de utilidad y funciones de software de aplicaciones.

Patrones de arquitectura

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular de la arquitectura, el cual es recurrente dentro de un cierto contexto, puede expresar una solución referente a uno o varios de los elementos que conformar las “4c” de la arquitectura: componentes, configuraciones, comunicaciones y restricciones. En este caso se especifica describiendo los componentes, con sus responsabilidades y relaciones.

Dentro de sus principales características se pueden citar[25]:

- Atacan problemas recurrentes que ocurren en situaciones específicas y dan una solución.

- Documentan experiencias de diseño existentes y bien probadas.
- Identifican y especifican abstracciones de alto nivel.
- Proveen un vocabulario común y comprensible.

2.9.1 **Propuesta arquitectónica**

El estilo a utilizar en el plugin es el Llamada – Retorno con el patrón arquitectónico Modelo-Vista-Controlador (MVC). El MVC permite separar los datos de la aplicación en tres componentes distintos: la interfaz de usuario, lógica del negocio y los datos de la misma.

El modelo contiene las clases relacionadas con el acceso a datos. Es importante que el código sea lo más genérico posible y que además se pueda reutilizar en otras situaciones y proyectos. No se incluye lógica en el modelo, solamente consultas a la base de datos y validaciones de entrada de datos.

La vista contiene las clases que representan la parte que será visualizada en pantalla por el usuario.

El controlador es la entrada a la aplicación, se mantiene a la escucha de todas las peticiones, ejecuta la lógica de la aplicación y muestra la vista apropiada para cada caso.

XP no establece ningún artefacto para la descripción arquitectónica sin embargo es uno de los elementos sin los cuales no se podría tener una solución consistente. La solución que se propone desde el punto de vista de la arquitectura se presentará a continuación mediante la vista arquitectónica, que es homóloga con la vista lógica, una de las 4+1 vistas que propone Kruchten [26] para la descripción de la arquitectura y que se utiliza en otras metodologías de desarrollo. No es más que la expresión visual del estilo y el patrón arquitectónicos seleccionados.

2.9.2 **Tarjetas Clase-Responsabilidades-Colaboración (CRC)**

El uso de las tarjetas C.R.C (Clase-Responsabilidad-Colaboración) permite al programador centrarse y apreciar el desarrollo orientado a objetos Las tarjetas CRC se describen en tres aspectos:

- Clase: representa una colección de objetos similares
- Responsabilidades: se le llama a lo que la clase sabe o hace. En ocasiones la clase no tiene toda la información para lograr determinados propósitos, lo que hace que deba interactuar con otras

- Colaboración: toma dos formas, un pedido de información o un pedido a que se realice una acción

Durante el proceso de diseño se elaboraron 6 tarjetas CRC. La figura 6 es una muestra de ellas. Puede consultarse el resto en el artefacto Tarjetas CRC, del proyecto.

Tarjeta CRC Clase Tabla

Tarjeta CRC	
Clase: Tabla	
Responsabilidades	**Colaboraciones**
✓ Nombre	MigraTablaMatch
✓ Nombre de los Campos	
✓ Tipos de campos	
✓ Campo	

Tabla 6. Tarjeta CRC Tabla

2.9.3 **Diagrama de clases**

Un diagrama de clases es un diagrama estático que se utiliza para modelar la vista estructural de un sistema. Describe gráficamente las especificaciones de las clases (atributos y métodos) además de visualizar las relaciones entre estas.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación.[27] Las relaciones que se establecen entre las tarjetas CRC se evidencian en el diagrama de clases

Vista arquitectónica

En el plugin se evidencia claramente el patrón Modelo-Vista-Controlador. En el Modelo se tienen las BD de Microsoft Access y PostgreSQL de las cuales se van a obtener y migrar los datos respectivamente. En la vista se encuentra la clase interfaz Widget que se va a encargar de mostrar al usuario los resultados de sus peticiones. El controlador, por su parte, ostenta las clases Migra_bd, Tabla, Analiza_bd y MigraTablaMatch, cuyas responsabilidades esencialmente radican en las operaciones de control.

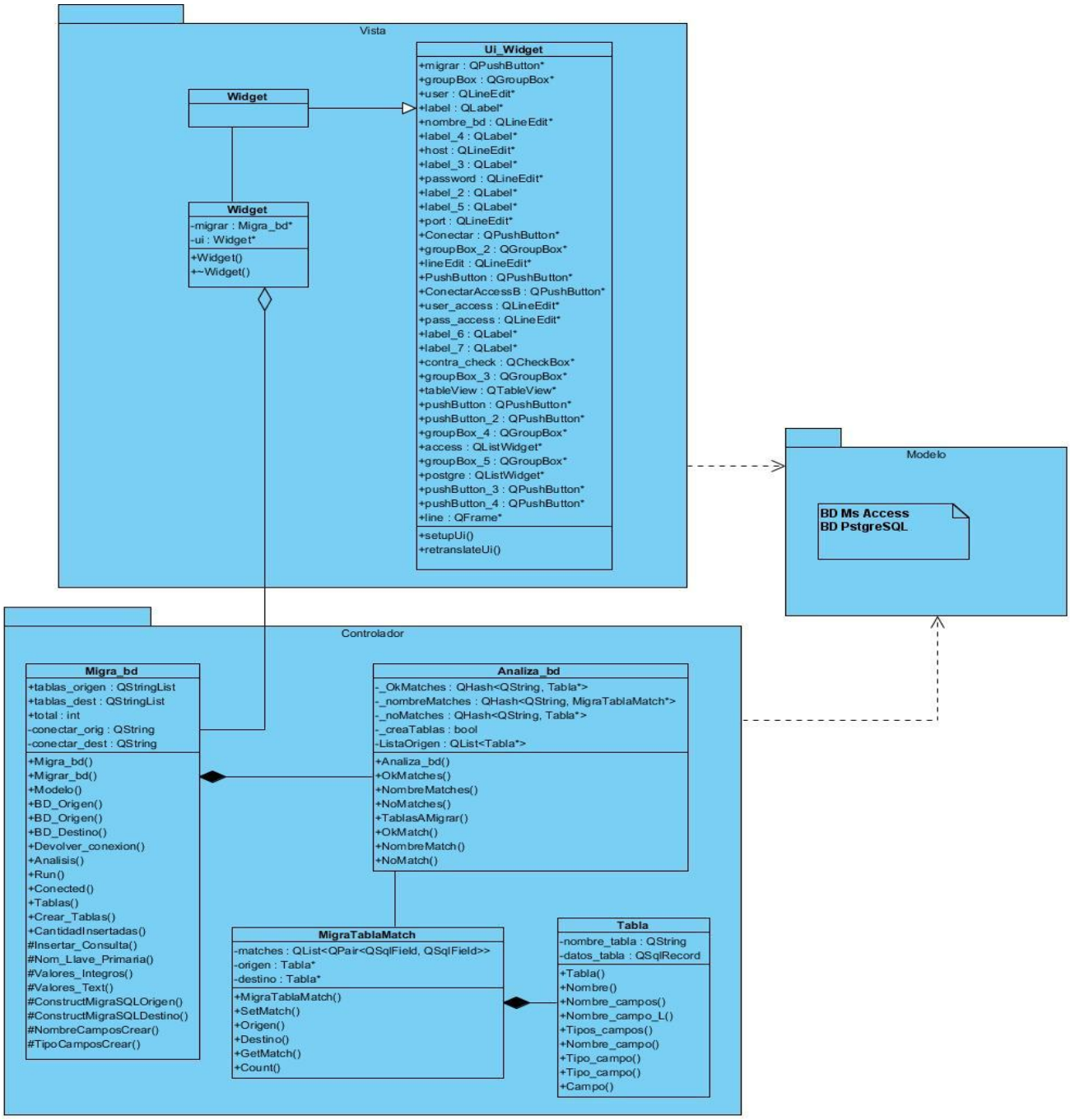

Figura 2.Vista Arquitectónica (con Tarjetas CRC)

Clases utilizadas

Para profundizar en las clases diseñadas, a continuación se profundiza en el papel de las cuatro principales.

Widget

Es la clase interfaz encargada de mostrar las peticiones por el usuario. Contiene los campos y botones necesarios para establecer la conexión a las BD. Visualiza la estructura y los datos de la tabla seleccionada. Brinda la posibilidad de elegir las tablas que se desean migrar. Las acciones se sellan con mensajes, satisfactorios o no, que ubican al usuario para dar el siguiente paso. Su importancia radica en ser mediadora entre el usuario y la lógica de la aplicación.

Migra_bd

Es una clase controladora, encargada de dirigir y realizar el proceso migración. Permite establecer las conexiones con las BD de origen y destino. Es la clase fundamental dentro del proceso.

Analiza_bd

Es una clase pensada para analizar las BD origen y destino y obtener de ellas la información de sus registros. Es relevante en la BD origen para estar al corriente de los registros posibles a migrar, y en el destino para no insertar una tabla ya existente.

MigraTablaMatch

Es la encargada de comprobar que una tabla en la base de datos de origen no sea igual a otra en el destino lo que evitará perder tiempo de ejecución del plugin.

2.9.4 **Patrones de diseño**

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Cabe destacar que existen varios grupos de patrones de diseño, pero la solución se basa en los más usados en función del objetivo que se ha planteado en este trabajo.

Patrones GRASP

Se les denomina estos patrones, GRASP, por las siglas en inglés de patrones generales de asignación de responsabilidades. Se basan en la determinación de las clases adecuadas y decidir cómo estas clases deben interactuar. Incluso cuando se utilizan metodologías rápidas como XP y el

proceso se centra en el desarrollo continuo, es necesario elegir cuidadosamente las responsabilidades de cada clase desde la primera codificación y, fundamentalmente, en la refactorización del programa. Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. [27] A continuación se presentan los patrones GRASP empleados y su concreción en el diseño.

Patrón Alta cohesión: Asigna a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad, evitando así que una clase sea la única responsable de muchas tareas en áreas funcionales muy heterogéneas.

Patrón Bajo acoplamiento: Se asignan las responsabilidades de forma tal que cada clase se comunique con el menor número de clases, minimizando el nivel de dependencia. A simple vista se puede observar que el diseño propuesto respeta este patrón al evidenciar clases con el menor número de clases relacionadas posible.

Ambos patrones (alta cohesión y bajo acoplamiento) están muy relacionados y pudiera decirse que tienen efectos visuales contrarios, sin embargo, la idea es que busque el equilibrio entre la aplicación de ambos. Un ejemplo del primero se evidencia en la clase Migra_bd responsable de realizar las funcionalidades necesarias para la migración, que para poder realizarlas necesita de otras clases. Las clases que intervienen en el proceso de migrar no se relacionan entre sí, sino que se relacionan directamente con la clase Migra_bd, poniendo en práctica el patrón bajo acoplamiento.

Patrón Creador: Este patrón ayuda a responder la pregunta de qué clase debe ser la responsable de crear nuevos objetos (instancias de alguna clase). Se manifiesta cuando a la clase Migra_bd se le asigna la responsabilidad de crear una instancia de la clase Analisis_db.
Controlador: Asigna la responsabilidad del manejo de los eventos del sistema a una clase. Este patrón se evidencia en la clase Migra_bd, la cual es la encargada de controlar y manejar todo el proceso de migración.

Patrón Experto: Asigna una responsabilidad al experto en información: se emplea cuando la clase que cuenta con la información necesaria para cumplir la responsabilidad, favorece la robustez y el fácil mantenimiento del sistema. En el plugin la clase controladora requiere de información distribuida en varias clases. Se evidencia en la clase controladora Migra_bd, pues contiene toda la información necesaria para realizar operaciones relacionadas con Tabla, MigraTablaMatch y Analiza_db.

Patrones GOF

GOF es la abreviación del grupo de los cuatro en inglés, compuesto por Erich Gamma, Richard Helm, Ralph Jhonson y John Vlisides, quienes en su publicación “Design Patterns” (década de los 90s), describen 23 patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño usando modelado con UML. Estos patrones se agrupan en las siguientes categorías: creacionales, estructurales y de comportamiento.[28]

Los patrones GOF de Comportamiento facilitan y definen la comunicación e interacción entre los objetos de un sistema y estando lo menos entrelazados posible. De los patrones que se encuentran en este grupo se utilizó el observador

El patrón observador tiene la responsabilidad de actualizar las dependencias de un objeto al cambiar el estado del mismo. Este patrón QT lo implementa por defecto, y se manifiesta frecuentemente entre las interfaces, cuando emiten señales. Utiliza el mecanismo de la interacción con las clases visuales mediante las señales y los Slot, donde de forma asincrónica los objetos que están suscritos a esa señal se enteran del cambio de estado.

2.10 **Conclusiones Parciales**

La descripción de la solución hasta su diseño a través de artefactos de la Metodología XP permitió una mejor comprensión de la propuesta. Se presentaron las funcionalidades para poder realizar la migración de las cuales solo una posee muy alta criticidad y mayor peso, pues deriva el grueso de los métodos que permitirán que los datos se migren. La descripción de estos requisitos mediante las HU permitió determinar los hilos conductores del proceso. Estas funcionalidades se expresaron en un diseño bien estructurado de clases, basado en patrones GRASP, lo que garantiza en buena medida que sea robusto y escalable. Se describió la arquitectura, basada en el patrón arquitectónico M-V-C, estableciendo límites entre la aplicación y los datos, importante elemento que garantizará independencia de la BD que se obtenga.

3

CAPÍTULO 3 IMPLEMENTACIÓN Y VALIDACIÓN

Introducción del capítulo

Definidas las bases estructurales se presentan los elementos de asociados a la implementación como el algoritmo fundamental empleado para migrar y el estándar de codificación. Siempre que se obtenga un sistema informático es necesario que se determine si cumple con las funcionalidades establecidas al inicio del proceso de desarrollo, permitiendo determinar en buena medida su calidad. Con esto se reduce el número de errores así como el tiempo entre la aparición de estos y su detección. En el presente capítulo se determinará la mejor manera de realizar las pruebas así como los métodos a emplear y los resultados.

2.11 **Algoritmo *Migrar***

El algoritmo de migración se basa hacer un mapa de la BD de origen, compararla con el destino y mediante estructuras de datos como las listas, hacer las transformaciones en términos de mapeado de datos y llevarlas a la BD destino

Pasos del algoritmo

- Se establece la conexión a MS Access
- Se obtienen los nombres de las tablas y de sus campos con sus tipos
- Se reconoce la llave primaria de cada tabla, con el objetivo de mantenerla en PostgreSQL.
- Se establece la conexión a PostgreSQL
- Se seleccionan las tablas a migrar que desee el usuario
- Se comparan las BD origen y destino
- Si la tabla ya existe tal como está en el origen no se crea en PostgreSQL
- Caso contrario se crea la tabla con los campos ya adquiridos en el paso 2

- Se señala como la llave primaria de la nueva tabla creada, la obtenida en el paso 3
- Se seleccionan las tuplas de una tabla, en la base de datos origen
- Se van insertando en el destino cada una de las tuplas, hasta conformar la tabla
- Se cierra la conexión

Para el desarrollo del algoritmo no se tuvo en cuenta la estructura de la BD de datos (llaves, foráneas, relaciones) pues el objetivo es solo migrar los datos.

2.12 **Estándares de codificación**

Una de las 12 prácticas de XP según Beck Kent es el establecimiento de estándares de codificación. Por lo cual se describe el estándar utilizado para la implementación de la solución, con ejemplos evidentes del código.

Identación

La unidad de identado es de 4 espacios evitando el uso de la tabulación.

```
BoolMigraTablaMatch::setMatch(constQString&campoOrigen)
{
    if(_origen->nombre_campos().count >nombre_campos().count())
{
    return true;
}
else
{
    return false;
}
}
```

Longitud de la Línea

Evitar líneas con más de 80 caracteres. Cuando una sentencia sobrepase una línea simple del editor, esta deber ser separada en otras. La separación debe hacerse preferentemente luego de una coma o

después de un operador. Realizar la ruptura después de un operador disminuye la probabilidad de que al realizar un copiado del código se cometa un error por olvido de la inserción del punto y coma. La siguiente línea debe ser identada con 5 espacios.

```
for (int i = 0 ; i<tabla->nombre_campos().count() ; i++)
{
   valores<<this->Valores_Integros(TipoCamposCrear(tabla->campo(i)),
   OrigenQuery.value(i).toString());
}
```

Comentarios

Es conveniente dejar información que pueda ser leída tiempo después por personas que necesiten entender qué fue lo que se hizo en el fragmento de código. Los comentarios deben ser escritos de forma simple y concisa, generalmente en una sola línea.

```
QStringmigra_bd::Nom_Llave_Primaria(constQstringtabla)//Nombredelcampollave
{
QSqlDatabaseAccess=Devolver_conexion("access");//access
QSqlTableModel*modelo=newQSqlTableModel(this,Access);
modelo->settable( table );
returnmodelo->primaryKey().fieldName(0);
}
```

Declaración de variables

Cada variable debe de ser declarada en una línea y comentada. También deben aparecer ordenadas alfabéticamente:

El nombre de las variables se escribe con minúsculas.

```
QSqlRecord datos_tabla; //Recoge lo datos de la tabla
QString nombre_tabla;   //Nombre de la tabla a analizar
```

Declaración de funciones

No debe haber espacio entre el nombre de la función y el paréntesis izquierdo, ni entre este y la lista de parámetros. Debe haber un espacio entre el paréntesis derecho y la llave de comienzo del cuerpo de la función. Las sentencias del cuerpo deben estar en la línea siguiente. La llave que cierra debe estar alineada con la línea de declaración de la función. Los nombres de las funciones se rigen por las mismas características que el de las variables con las palabras claves dentro del mismo comenzando por mayúscula.

```
QStringmigra_bd::Nom_llave_Primaria(const Qstring tabla)//Nombredelcampollave
{
QSqlDatabase Access = Devolver_conexion("access");//access
QSqlTableModel  *modelo = new QSqlTableModel(this,Access);
modelo->setTable(tabla);
return modelo->primaryKey().fieldName(0);
}
```

Identificadores

Los identificadores pueden estar formados por cualesquiera de las 26 letras del alfabeto inglés, minúsculas o mayúsculas (A .. Z, a .. z), los 10 dígitos (0 .. 9) y el carácter subrayado "_". Debe evitarse el uso de caracteres internacionales (ej.: ñ, ü) porque no siempre pueden ser leídos o entendidos correctamente en todos los lugares. No se debe usar el símbolo dólar "$" o la barra invertida "\" en los identificadores.

Sentencias

Sentencias Simples

Cada línea debe contener como máximo una sentencia. Se debe poner un punto y coma ";" al final de cada sentencia simple. Una sentencia de asignación puede resultar en la asignación de una función o de un objeto como literal y en todos los casos como sentencia de asignación debe estar finalizada con un punto y coma.

Sentencias Compuestas

Las sentencias compuestas son aquellas que contienen una lista de sentencias encerradas entre llaves:
Las sentencias encerradas serán identadas a 4 espacios.

La llave que inicia la lista de sentencias debe estar al final de línea de la sentencia compuesta.

La llave que termina la lista de sentencias debe estar al comienzo de una línea y guardar la misma identación que la sentencia compuesta en correspondencia con la llave que inicia.

Las llaves siempre serán usadas para listar todas las sentencias, aunque se trate de una sola, cuando son parte de una estructura de control como *if* o *for.* Ello facilita agregar nuevas sentencias sin la introducción accidental de errores.

Etiquetas

Las sentencias etiquetadas son opcionales. Solo estas sentencias de ben ser etiquetadas: while, do, for, foreach, switch.

```
Foreach (QString sentence,lote)
{
Query = Pgsql.exec(sentence);
}
```

Sentencia return

Una sentencia return no debe utilizar paréntesis “()” alrededor del valor que se retorna. La expresión cuyo valor se retorna debe comenzar en la misma línea que la palabra reservada return, terminada con un punto y coma.

```
…
{…
    Return campos;
 …
}
```

Sentencia if

La sentencia if debe ser escrita de esta manera:

```
if (condition)
{
```

```
    sentencia
}
if (condición)
{
    sentencia
} else
{
    sentencia
}
if (condición)
{
    sentencia
}
else if (condición)
{
    sentencia
}
else
{
    sentencia
}
```

Estructuras repetitivas

Las estructuras repetitivas deben ser escritas de esta manera:

```
for (inicialización; condición; actualización)
{
    sentencia
}
while (condición)
{
    sentencia
}
foreach (valor1, valor2)
{
    sentencia
}
```

Sentencia switch

La sentencia switch debe ser escrita de la siguiente forma:

```
switch(campo.type())
{
   Case QVariant::Double:
    return  “float”;
   default:
    return “text”;
}
```

Espacios en blanco

Las líneas en blanco facilitan la lectura determinando secciones de código lógicamente relacionada. Los espacios en blanco pueden ser (o no deben ser) utilizados en las siguientes circunstancias:

- No se debe utilizar un espacio en blanco entre el identificador de una función y el paréntesis que abre a la lista de parámetros. Ello ayuda a distinguir entre palabras reservadas y llamadas a funciones.
- No se debe utilizar el espacio para separar un operador unario de su operando excepto cuando ese operador es una palabra como typeof.
- Debe dejarse un espacio luego de cada coma “,” y signo de igualdad“=”.

Declaraciones de clases

Solo debe existir un fichero con más de una clase declarada.

```
#include <QSqlField>
#include “tabla.h”

Tabla::Tabla(const Qstring &nombre, const QSqlRecord &datos)
{
   nombre_tabla = nombre;
   datos_tabla  =datos;
```

```
}
QSqlFieldTabla::campo(const QString &nombre)const
{
   Return datos_tabla.field(nombre);
}
```

2.13 **Interfaz del plugin**

A continuación se muestra la interfaz del plugin, donde se establecen las conexiones a las BD MS Access y PostgreSQL. Se divide en siete partes:

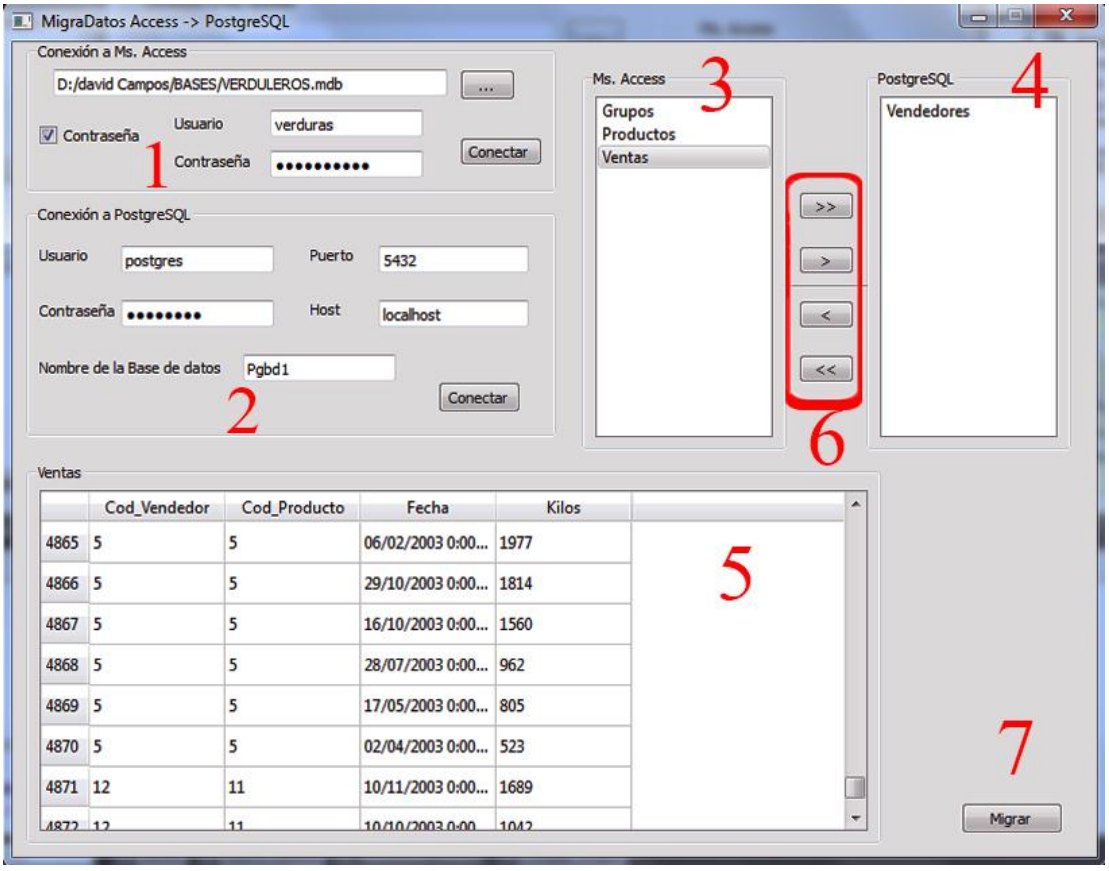

Figura 3 Interfaz principal

1. Se encarga de la conexión a MS Access. Da la posibilidad de seleccionar el archivo origen, y conectarse, además de especificar usuario y contraseña. Si se conecta envía un mensaje satisfactorio, en caso contrario se envía un mensaje con el tipo de error.

2. En esta área se definen los parámetros de conexión a PostgreSQL, además se establece la propia conexión, se muestra mensaje de confirmación.

3. Una vez establecida la conexión con Access, en esta área se listarán los nombres de las tablas de la BD MS Access.

4. Cuando se conecta a PostgreSQL se habilita esta zona, en la que se seleccionan las tablas a migrar.

5. Este cuadro permite mostrar los datos de una tabla seleccionada de la BD origen.

6. Son botones que permiten seleccionar las tablas que se desean migrar. Hay botones que trasladan las tablas para Access por si ocurre una equivocación por parte del usuario.

7. El botón migrar, posibilita la migración de las tablas seleccionadas por el usuario que se encuentran en la zona 4.

2.14 **Verificación y validación vista desde XP**

Para XP la corrección va de la mano de la codificación, que es lo que se conoce como codificar y corregir, una práctica muy común en esta metodología. Las pruebas son la forma de expresar la verificación en esta metodología siendo una de las doce prácticas más frecuentes en cada una de sus fases según Kent Beck.

Las pruebas de software no son más que un proceso mediante el cual se ejecuta el software con el objetivo de identificar fallas o errores. En XP se deben realizar de forma continua a lo largo del proyecto fundamentalmente luego de cada iteración y se dividen en dos tipos: las pruebas unitarias y las pruebas de aceptación o funcionales.

2.15 **Pruebas a emplear para verificar y validar el plugin**

Para la verificación de la solución propuesta respetando lo establecido por la metodología de desarrollo empleada, se realizan las pruebas de aceptación. Mediante estas pruebas se evaluará el cumplimiento de las funcionalidades establecidas al concluir cada iteración y las pactadas con el cliente.

2.15.1 **Pruebas aceptación**

Las pruebas de aceptación permiten al cliente saber cuándo el sistema funciona, y que los programadores conozcan qué es lo que resta por hacer.[29] Son de las pruebas que más comúnmente se utilizan en esta metodología de desarrollo.

Las pruebas de aceptación son pruebas de caja negra definidas por el cliente para cada HU y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Estas pruebas corresponden a una especie de documento de requerimientos en XP, ya que marcan el camino a seguir en cada iteración, indicándole al equipo de desarrollo hacia dónde tiene que ir y en qué puntos o funcionalidades debe poner el mayor esfuerzo y atención.[30]

Estas pruebas se basan en las necesidades del cliente, pero son su responsabilidad pues no son especialistas y carecen de tiempo. Debe ser el probador el que escriba los casos de prueba, interpretando las ideas del cliente, basándose en sus criterios de estabilidad y rendimiento. Estas pruebas no exigen 100% de efectividad.

La aplicación de las pruebas de aceptación se basa en la descripción de casos de pruebas desde el punto de vista del cliente. Los casos de pruebas exitosos en una iteración deben repetirse de forma satisfactoria en las siguientes. Un error en un paso hace que se determine que el caso de prueba falló.

En el caso que se presenta será tomado el plugin como un módulo en sí, en el que la posibilidad de fallar viene a estar dada, fundamentalmente, por el cumplimiento de las funcionalidades establecidas por el cliente. Aunque los dos tipos de pruebas (unitarias y de aceptación) son complementarias, el

trabajo se basa en pruebas que se especializan en la verificación de las funcionalidades pactadas que son las de aceptación, donde el cliente juega un papel esencial como parte del equipo de desarrollo, y donde el 100% del producto debe estar correcto.

2.16 **Método de prueba Caja negra**

El método Caja negra en XP se basa en el establecimiento de casos de prueba en función de las HU contenedoras de los requisitos funcionales. Para desarrollar las prueba de caja negra existen varias técnicas, entre las que están[23]:

- Técnica de la Partición de Equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica de la Partición de Equivalencia es una de las más efectivas permitiendo chequear los valores válidos e inválidos de las entradas existentes en el software, identificando de forma rápida una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se centra en la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices[23]:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen tres clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor concreto, aparecen tres clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen dos clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay dos clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases. Para definir las clases de equivalencia hace falta tener en cuenta un conjunto de reglas:

- Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos inválidas.
- Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase inválida.
- Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una inválida.
- Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

2.17 **Casos de prueba**

A continuación se presenta una muestra de Caso de prueba para la HU “Conectar a base de datos PostgreSQL destino”, que se basa en la regla expuesta en el epígrafe anterior en cuanto a la determinación de las clases equivalentes.

Caso de prueba aplicado a la Historia de Usuario “Conectar a base de datos PostgreSQL destino”.

		Variables						
Escenario	**Descripción**	**Var. 1**	**Var. 2**	**Var. 3**	**Var. 4**	**Var 5**	**Respuesta del sistema**	**Flujo central**
EC1. Se establece conexión	Se establece la conexión definida la BD destino	V	V	V	V	V	El sistema se conecta a la BD especificada	1. El usuario escribe los parámetros necesarios para establecer la conexión correctamente. 2.Selecciona la opción conectar 3.El sistema muestra mensaje de conexión satisfactoria
EC2.No se establece conexión	No se establece conexión por insertarse datos no válidos. Los no válidos incluyen los vacíos	I	V	V	V	V	El sistema no se conecta a la BD especificada y muestra mensaje de error	1. El usuario escribe los parámetros necesarios para establecer la conexión. 2. Selecciona la opción conectar. 3. Muestra
		V	I	V	V	V		
		V	V	I	V	V		
		V	V	V	I	V		
		V	V	V	V	I		
		I	I	V	V	V		
		I	V	I	V	V		
		I	V	V	I	V		
		I	V	V	V	I		
		V	I	I	V	V		
		V	I	V	I	V		
		V	I	V	V	I		
		V	V	I	I	V		

		V	V	I	V	I		mensaje de error que especifica la existencia de parámetros no válidos
		V	V	V	I	I		
		I	I	I	V	V		
		I	I	V	I	V		
		I	I	V	V	I		
		I	V	V	I	I		
		I	V	I	I	V		
		V	I	I	I	V		
		I	V	I	V	I		
		I	I	I	I	V		
		I	I	I	V	I		
		I	I	V	I	I		
		I	V	I	I	I		
		V	I	I	I	I		
		I	I	I	I	I		

Tabla 7 Caso de prueba Conectar a base de datos PostgreSQL

2.18 **Resultados**

Además de estas pruebas se empleó una BD en MS Access de prueba que contiene los tipos de datos que fueron mapeados en las etapas de la migración para probar la efectividad del proceso realizado arrojando que se correspondieron los resultados obtenidos con los esperados. Se probó el sistema como un plugin más de HABD con resultados satisfactorios.

Después de haber sido aplicadas las pruebas a partir de los casos de pruebas propuestos al concluir la primera iteración según el plan de iteraciones y de acuerdo a las funcionalidades implementadas hasta la fecha se detectaron siete no conformidades, que fueron resueltas en un período de tiempo de dos días. Se realizó una segunda iteración y se detectaron dos no conformidades siendo solucionadas en tres horas, se realizó otra corrida detectándose una no conformidad, y luego se probó otra vez con 100% de los resultados esperados. Las no conformidades fueron resueltas luego de identificadas. Lo anterior se evidencia en la figura siguiente.

No Conformidades detectadas, significativas y no significativas por iteración.

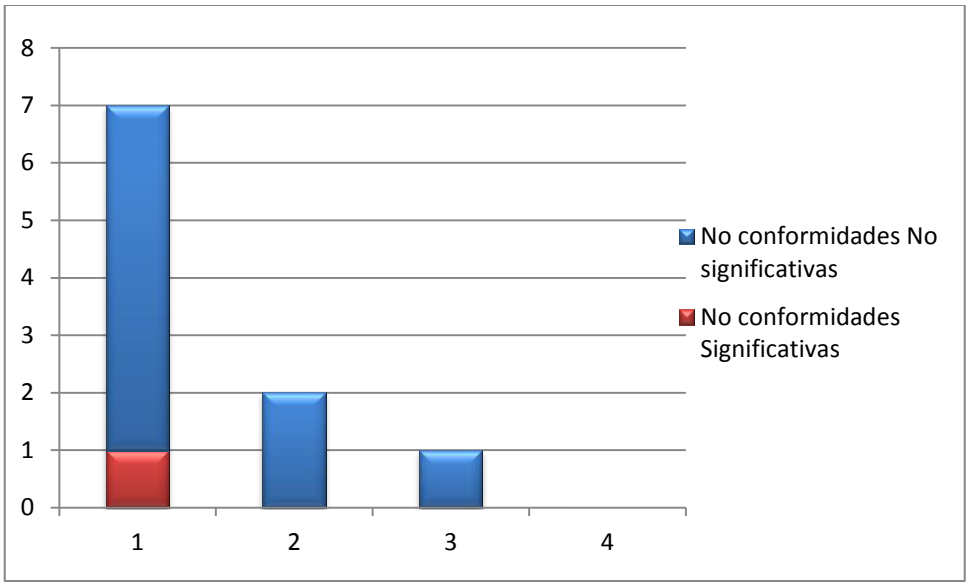

Figura 4. No Conformidades significativas y no significativas detectadas por iteración

2.19 **Conclusiones Parciales**

En este capítulo se presentó el principal algoritmo empleado para la migración, lo que permitió un análisis lógico para la obtención de la solución satisfactoria. Luego de haber hecho una breve presentación de las pruebas vistas desde la metodología XP, se optó por la realización de pruebas de aceptación con las que se validó, por las características propias de la solución, aplicándose el método Caja negra. Los casos de prueba partiendo de las HU, tuvieron en cuenta los puntos de vistas del cliente siempre respetando sus condiciones. Las pruebas se realizaron en cuatro iteraciones hasta obtener el 100% de los resultados esperados.

CONCLUSIONES

Con este trabajo de manera general se obtuvo un plugin que realiza la migración de datos desde MS. Access a PostgreSQL capaz de integrarse a HABD permitiendo que sea esta una herramienta más completa para los administradores de BD PostgreSQL. Este resultado principal se basó en que:

- Se establecieron los criterios teóricos bajo los cuales se realizó la migración de datos de MS Access a PostgreSQL de acuerdo al ajuste realizado las etapas propuestas para este proceso
- Se diseñó un algoritmo que garantizó la migración satisfactoria que puede emplearse para la migración de datos entre otros gestores con adecuaciones mínimas. El algoritmo se implementó de conjunto con el de las de las funcionalidades previstas siguiendo la metodología de desarrollo XP.
- Las validaciones efectuadas mediante las pruebas con los criterios establecidos garantizan que los resultados de la migración son fieles a los esperados.

El sistema contribuirá a la soberanía tecnológica del país, facilitando la migración datos de las BD Access a PostgreSQL mediante HABD, aportando en términos de integralidad.

RECOMENDACIONES

- Se recomienda para futuras versiones migrar además de los datos que fue el objetivo de este trabajo, la estructura, por lo que sería necesario incorporar las restricciones “Foreign key”, “Check”, “Unique” y “Not null” a los datos migrados.

REFERENCIAS BIBLIOGRÁFICAS

1. VIÑOLO SOSA, R. R. Y. R. F., ALEXANDER. Sistema Gestor de Procesos de Media v2. Serie Científica de la Universidad de las Ciencias Informáticas [Type of Work]. 2012, vol. 5, 2013]. Available from:<http://publicaciones.uci.cu/index.php/SC/article/view/967>.

2. DATE, C. J. AND S. L. R. FAUDÓN Introducción a los sistemas de bases de datos. Edtion ed.: Pearson Educación, 2001. ISBN 9789684444195.

3. ROZIC, S. E. Bases de Datos y su Aplicacion con SQL. Edtion ed.: Mp Ediciones Corporation, 2004. ISBN 9789875262133.

4. ALE, M.-. Introducción a las bases de datos relacionales [online]. [Buenos Aires, Rep. Argentina]: 2000 [cited 3 de diciembre 2012].

5. MALAGA, U. Tipos de Bases de Datos [online]. [Malaga]: 2010 [cited 01/02 2013]. Available from:<http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf>.

6. MATO GARCÍA, L. R. M. Diseño de Bases de Datos [online]. 1999.

7. ACOSTA, I. A. Guía para la Migración de datos en Access. 2009, [cited 10 de diciembre 2012]. Available from:<http://postgresql.uci.cu/>.

8. MARTÍN PÉREZ, C. J. POSTGRESQL [online]. 19. 2010 [cited 8/12 2012]. Available from:<http://softwarelibrecr.org/sites/default/files/articulo.pdf>.

9. Almaguer , J. and A. Camué, *Plugin de recuperación del estado de entidades para la herramienta de administración de bases de datos HABD*, in *DATEC*. 2012, Universidad de las Ciencias Informáticas: La Habana.

10. Gámez Gutiérrez, Y. and R. Almanza Vázquez, *Migración de los datos manejados por la Dirección de Cooperación Internacional hacia el Sistema de Gestión de Cooperación Internacional.* 2012, UCI.

11. Estrada, Y. and R. Fonseca, *Sistema para la Migración y Conversión de Datos (SIMIC-D).en la Facultad Regional de la Universidad de las Ciencias Informáticas de Ciego de Ávila*. 2012, UCI: Ciego de Ávila.

12. SCRIBD. *Metodologias de desarrollo software.* 2008 [cited 2012 18 de diciembre]; Available from: http://www.scribd.com//doc/2050925/metodologias-de-desarrollo-software.

13. MÉNDEZ, A. V. Metodología de desarrollo de software [online]. [Michoacán]: Institución Tecnológico Superior de Apatzingán, 2010 [cited 18 de diciembre 2012].

14. RUMBAUGH, J., G. BOOCH, I. JACOBSON, H. C. RODRÍGUEZ, et al. El Lenguaje unificado de modelado: manual de referencia. Edtion ed.: Pearson Educación, 2007. ISBN 9788478290871.

15. VISUAL-PARADIGM. Visual Paradigm Internacional. [online]. 2010 [cited 20/02 2013]. Available from:<http://www.visual-paradigm.com/product/vpuml/>.

16. PRATT, T. W., M. V. ZELKOWITZ AND H. J. E. GARCÍA Lenguajes de programación: diseño e implementación. Edtion ed.: Prentice-Hall, 1998. ISBN 9789701700464.

17. MEYER, L. D. AND P. NICANOR. Introducción al desarrollo multiplataforma con Qt 4. 2007, Access 2007]. Available from: <www.perezmeyer.com.ar/files>.

18. SALAVERT, I. R. AND M. D. L. PÉREZ Ingeniería del software y bases de datos: tendencias actuales. Edtion ed.: Universidad de Castilla-La Mancha, 2000. ISBN 9788484270775.

19. UNVIERSIDAD MURCIA, D. I. Guión de prácticas, Programación visual con Qt Creator. 2010 vol. 1, p. 30 Access 2010]. Available from: <http://dis.um.es/~ginesgm/files/doc/pav/guion1.pdf>.

20. LABCOM. Tema3 Modelos de Dominio [online]. 2011 [cited 20/02 2013]. Available from:<http://www.labcom.upcomillas.es/isw2/apuntes/01Tema3-Modelodedominio.pdf>.

21. MORA, M. A. Historias de usuario [online]. 2010 [2013]. Available from:<http://www.slideshare.net/MiquelMora/historias-de-usuario>.

22. Rodríguez Rojas, E. and Y. López Pérez, *Plugin CRUD-PG para la herramienta de administración de bases de datos HABD.* 2012, UCI: La Habana.

23. PRESSMAN, R. S. A. AND J. E. M. MURRIETA Ingeniería del software: un enfoque práctico. Edtion ed.: Mcgraw Hill/Interamericana Editores, 2006. ISBN 9789701054734.

24. NAVA, M. *Arquitectura de Software Sistemas de Llamada y Retorno*. 2011 [cited 2013; Available from: http://es.scribd.com/doc/23161581/Estilos-Arquitectonico.

25. OCHOA, P. S. Introducción a los Patrones. Diseño y Arquitectura [online]. 2005 [cited 26/02 2013]. Available from:<www.u-cursos.cl/ingenieria/2005/2/CC51A/1/material_docente/>..

26. KRUCHTEN, P. El Modelo de “4+1” Vistas de la Arquitectura del Software.[cited 15/04 2013], pp.16. Available from:<http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf>.

27. Larman, C., *UML y patrones: introducción al análisis y diseño orientado a objetos*. 2004: Félix Varela.

28. HERNÁNDEZ, P. V. Uso de Patrones de Arquitectura. 2007. Available from:<es.scribd.com/doc/.../Uso-de-Patrones-de-Arquitectura-Capitulo-4>.

29. RON, J. Extreme Testing [online]. 2005 [cited 20/01 2013]. Available from:<http://www.xprogramming.com/publications/SP99ExtremeforWeb.pdf>.

30. DAYVIS, M., C. DIEGO, C. FERNANDO, C. J. PABLO, et al. Testing en eXtreme Programming. Gestión de Software 2006 [Type of Work]. 2006, [cited 22/04 2013].

BIBLIOGRAFÍA

1. ACOSTA, I. A. Guía para la Migración de datos en Access. 2009, [cited 10 de diciembre 2012]. Available from:<http://postgresql.uci.cu/>.

2. ALE, M.-. Introducción a las bases de datos relacionales [online]. [Buenos Aires, Rep. Argentina]: 2000 [cited 3 de diciembre 2012].

3. ALMAGUER , J. AND A. CAMUÉ. Plugin de recuperación del estado de entidades para la herramienta de administración de bases de datos HABD. Ingeniería Universidad de las Ciencias Informáticas, 2012.

4. ANGELFIRE. Patrones GoF [online]. 2011 [cited 09/11 2012]. Available from:<http://geektheplanet.net/5462/patrones-gof.xhtml>.

5. BECK, K. Planning eXtreme programming. Edtion ed.: ADDISON WESLEY Publishing Company Incorporated, 2001. ISBN 9780201710915.

6. CANALES MORA, R. Patrones de GRASP [online]. 2013 [cited 02/03 2013]. Available from:<http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=grasp>.

7. CARRRILLO PÉREZ, I., PÉREZ GONZÁLEZ, RODRIGO Y RODRÍGUEZ MARTÍN, AURELIANO DAVID. Metodología de desarrollo de software [online]. 2008 [cited 18 de diciembre 2012].

8. CAPOTE, O. P. Introducción a las bases de datos: el modelo relacional. Edtion ed.: Thomson-Paraninfo, 2005. ISBN 9788497323963.

9. DATE, C. J. AND S. L. R. FAUDÓN Introducción a los sistemas de bases de datos. Edtion ed.: Pearson Educación, 2001. ISBN 9789684444195.

10. DAYVIS, M., C. DIEGO, C. FERNANDO, C. J. PABLO, et al. Testing en eXtreme Programming. Gestión de Software 2006 [Type of Work]. 2006, [cited 22/04 2013].

11. DRAKE, J. M. AND P. LÓPEZ. Verificacion y Validación. 2009. Available from:<http://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_Verificacion Validacion-2011.pdf>.

12. ESTRADA, Y. AND R. FONSECA. Sistema para la Migración y Conversión de Datos (SIMIC-D).en la Facultad Regional de la Universidad de las Ciencias Informáticas de Ciego de Ávila. UCI, 2012.

13. FIESTAS GUTIÉRREZ, M. D. L. Á. SGBD Comerciales vs Libres Slideshare [Type of Work]. 2012, [cited 8 de diciembre 2012]. Available from:<http://www.slideshare.net/maanfi16/sgbd-comerciales-vs-libres#btnPrevious>.

14. FIGUEREDO FALCÓN, Y. AND R. RODRÍGUEZ PAJARES. Plugin para el particionado de tablas en bases de datos PostgreSQL mediante la Herramienta de Administración de Bases de Datos HABD. UCI, 2012.

15. FUENTES PAREDES, C. E. AND P. H. VEGA SILVA. Sistema web de servicios musicales para la corporación musicológica ecuatoriana., 2009.

16. GÁMEZ GUTIÉRREZ, Y. AND R. ALMANZA VÁZQUEZ. Migración de los datos manejados por la Dirección de Cooperación Internacional hacia el Sistema de Gestión de Cooperación Internacional. Programación Universidad de las Ciencias Informáticas, 2012.

17. GUTIÉRREZ, J. J., M. J. ESCALONA, M. MEJÍAS AND J. TORRES. Pruebas de sistema en Programación Extrema. 2008, pp. 12. Available from:<www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf>.

18. HERNÁNDEZ, P. V. Uso de Patrones de Arquitectura. 2007. Available from:<es.scribd.com/doc/.../Uso-de-Patrones-de-Arquitectura-Capitulo-4>.

19. JIMÉNEZ, A. B. Desarrollo de un Software de Medición del Estrés para un Dispositivo Foto-pletismógrafo Basado en el Protocolo USB 2.0. Unidad Zacatenco, 2010.

20. KRUCHTEN, P. El Modelo de "4+1" Vistas de la Arquitectura del Software. [cited 15/04 2013], pp. 16. Available from:<http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:modelo4_1.pdf>.

21. LABCOM. Tema3 Modelos de Dominio [online]. 2011 [cited 20/02 2013]. Available from:<http://www.labcom.upcomillas.es/isw2/apuntes/01Tema3-Modelodedominio.pdf>.

22. LARMAN, C. UML y patrones: introducción al análisis y diseño orientado a objetos. Edtion ed.: Félix Varela, 2004.

23. LEÓN CARRILLO, L. V. Caracterización de la Prueba de Software: Clasificación y Técnicas. 2010, [cited 20/03 2012]. Available from:<http://www.e-quallity.net/articulos/SG-200505-Luis05.pdf>.

24. LETELIER, P. AND M. D. C. PNADÉS. Metodologías ágiles para el desarrollo de software , eXtreme Programing (XP). [online]. [Valencia]: 2006 [cited 21/02 2013]. Available from:<www.willydev.net/descargas/masyxp.pdf>.

25. MÁLAGA, U. Tarjetas CRC [online]. 2011 [cited 02/02 2013]. Available from:<http://lsi.ugr.es/~ig1/docis/crc.pdf>.

26. MÁLAGA, U. Tipos de Bases de Datos [online]. [Malaga]: 2010 [cited 01/02 2013]. Available from:<http://www.lcc.uma.es/~galvez/ftp/bdst/Tema2.pdf>.

27. MOMJIAN, B. PostgreSQL: introduction and concepts. Edtion ed.: Addison-Wesley, 2001. ISBN 9780201703313.

28. MARTÍN PÉREZ, C. J. POSTGRESQL [online]. 19. 2010 [cited 8/12 2012]. Available from:<http://softwarelibrecr.org/sites/default/files/articulo.pdf>.

29. MAS, R. Sobre PostgreSQL [online]. 2010 [cited 03/11 2012]. Available from:<http://www.postgresql.org.es/sobre_postgresql>.

30. MATO GARCÍA, L. R. M. Diseño de Bases de Datos [online]. 1999.

31. MORA, M. A. Historias de usuario [online]. 2010 2013]. Available from:<http://www.slideshare.net/MiquelMora/historias-de-usuario>.

32. NAVA, M. Arquitectura de Software Sistemas de Llamada y Retorno [online]. 2011. Available from:<http://es.scribd.com/doc/23161581/Estilos-Arquitectonico>.

33. OCHOA, P. S. Introducción a los Patrones. Diseño y Arquitectura [online]. 2005 [cited 26/02 2013]. Available from:<www.u-cursos.cl/ingenieria/2005/2/CC51A/1/material_docente/>.

34. PÉREZ CASTELLANOS, Y. AND M. A. MOREJÓN MARTÍNEZ. Plugin diseñador de bases de datos para la herramienta de administración de bases de datos HABD. UCI, 2012.

35. PRESSMAN, R. S. A. AND J. E. M. MURRIETA Ingeniería del software: un enfoque práctico. Edtion ed.: Mcgraw Hill/Interamericana Editores, 2006. ISBN 9789701054734.

36. RIGGS, S. AND H. KROSING PostgreSQL 9 Administration Cookbook: Solve Real-world PostgreSQL Problems with Over 100 Simple, Yet Incredibly Effective Recipes. Edtion ed.: Packt Pub., 2010. ISBN 9781849510288.

37. ROB, P. A. AND C. CORONEL Sistemas de bases de datos: diseño, implementación y administración. Edtion ed.: Cengage Learning Latin America, 2004. ISBN 9789706862860.

38. RODRÍGUEZ ROJAS, E. AND Y. LÓPEZ PÉREZ. Plugin CRUD-PG para la herramienta de administración de bases de datos HABD. UCI, 2012.

39. ROZIC, S. E. Bases de Datos y su Aplicacion con SQL. Edtion ed.: Mp Ediciones Corporation, 2004. ISBN 9789875262133.

40. UNVIERSIDAD MURCIA, D. I. Guión de prácticas, Programación visual con Qt Creator. 2010 vol. 1, p. 30 Access 2010]. Available from: <http://dis.um.es/~ginesgm/files/doc/pav/guion1.pdf>.

41. VÁZQUEZ MONTERO, T. AND D. A. LINARES BROOKS. Plugin “Editor de funciones y disparadores” para la herramienta de administración de bases de datos HABD. Desarrollo UCI, 2012.

42. VIÑOLO SOSA, R. R. Y. R. F., ALEXANDER. . Sistema Gestor de Procesos de Media v2. Serie Científica de la Universidad de las Ciencias Informáticas [Type of Work]. 2012, vol. 5, 2013]. Available from:<http://publicaciones.uci.cu/index.php/SC/article/view/967>.

43. VISUAL-PARADIGM. Visual Paradigm Internacional. [online]. 2010 [cited 20/02 2013]. Available from:<http://www.visual-paradigm.com/product/vpuml/>.