UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

“GUÍA REFERATIVA PARA EL DISEÑO DE BASE DE DATOS DEL SISTEMA DE GESTIÓN CEDRUX”

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

Autores:

Yisel Pérez Fernández

José Manuel Morejón Barceló

Tutor: Ing. Ariel Torres Gálvez

Co-Tutor: Ing. Leandro Pérez-borroto Vivero

Ciudad de la Habana, Junio de 2010

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de _____del año 2010.

Yisel Pérez Fernández	José Manuel Morejón Barceló
_____	_____
Tutor:	**Co-Tutor:**
Ing. Ariel Torres Gálvez	Ing. Leandro Pérez-borroto Vivero
_____	_____

DATOS DE CONTACTO

Nombre y Apellidos del Tutor: Ing. Ariel Torres Gálvez.

Email: atgalvez@uci.cu

Currículum: Ing. Ariel Torres Gálvez graduado de Ingeniero Informático en el año 2008 en la Universidad de las Ciencias Informáticas. Comienza a desempeñarse profesionalmente en la facultad 3 como profesor y posteriormente en la Dirección de Producción 4, donde actualmente ocupa el rol de arquitecto de datos del proyecto ERP-Cuba.

Nombre y Apellidos del Co-Tutor: Leandro Pérez-borroto Vivero.

Email: lvivero@uci.cu

Currículum: Ing. Leandro Pérez-borroto Vivero graduado de Ingeniero Informático en el año 2009 en la Universidad de las Ciencias Informáticas. Actualmente ocupa el cargo de arquitecto de infraestructura del proyecto ERP-Cuba.

SÍNTESIS

Con la tendencia actual a la informatización, las empresas se han dado a la tarea de automatizar todos los procesos posibles dentro de su negocio, facilitándoles el trabajo y brindándoles sin dudas una mayor y mejor organización, lo cual contribuye en gran medida a mejores toma de decisiones.

El Proyecto ERP-Cuba es el encargado del desarrollo de un Sistema Integral de Gestión que proporcione todo un conjunto de facilidades a las entidades que hagan uso del mismo. Por tal motivo, se hace necesario un diseño eficaz de su base de datos, la cual permita almacenar la información de manera organizada y que luego pueda ser encontrada y utilizada fácilmente.

El presente trabajo de diploma se centra en la elaboración de una guía referativa que garantice un buen diseño de base de datos para el Sistema Integral de Gestión Cedrux, tomando como punto de partida la falta de coherencia existente en los diseños de los distintos módulos que la componen. Esta guía incluye buenas prácticas para la obtención de un diseño exitoso, así como un conjunto de soluciones que la Subdirección de Tecnología conjuntamente con el Arquitecto principal de datos, especificó como importantes para el diseño de la base de datos de Cedrux.

El desarrollo de este trabajo fue guiado por las actividades establecidas y por la utilización de herramientas especificadas por la Subdirección de Tecnología tales como: herramientas de diseño, gestor de base de datos y pruebas de concepto sobre las soluciones propuestas.

ÍNDICE

INTRODUCCIÓN

Actualmente muchas empresas necesitan más que nunca tener una organización y mantener de forma eficiente grandes volúmenes de datos, además de una herramienta que les proporcione el control para poder administrarlos. Debido a esto, toda empresa tiene sus datos almacenados, utilizando para esto el Sistema Gestor de Base de Datos (SGBD) para su manipulación, sin esto resultaría imposible de tratar los datos en su totalidad y se perdería tiempo y dinero innecesario.

A lo largo de las década de los 60 cuando las bases de datos entraron por primera vez al mercado, los diseñadores de base de datos trabajaban de manera muy primitiva, y muchos confundían frecuentemente el diseño de base de datos con la implantación de la base de datos [1].

Esta situación ahora ha cambiado, mediante el avance de la tecnología, ha evolucionado el diseño y una buena base de datos no es algo que simplemente suceda, la estructura de su contenido debe diseñarse con cuidado. Incluso un buen Sistema Gestor de Base de Datos funcionará deficientemente si cuenta con una base de datos mal diseñada.

Por lo tanto, la razón para preocuparse por el diseño es crucial para la consistencia, integridad y precisión de los datos. Existiendo un buen diseño de base de datos se facilita la administración de los mismos, convirtiéndose la base de datos en un valioso generador de información.

Entre las metas más importantes que se persiguen con un buen diseño se encuentran: acceso eficiente a la información con redundancia mínima, facilidad para extraer la información requerida, además de la velocidad de acceso y el comportamiento del manejador de bases de datos con cada tipo de información.

Desafortunadamente, las guías de diseño de datos no son muy populares, la mayoría de las organizaciones y de los diseñadores individuales confían muy poco en estas para llevar a cabo el diseño y esto se considera, con frecuencia, una de las principales causas de fracaso en el desarrollo de los sistemas de información. Debido a la falta de enfoques estructurados para el diseño de bases de datos, las mismas son inadecuadas o ineficientes, y el mantenimiento es difícil [1].

El mal diseño tiende a generar errores que probablemente conduzcan a toma de decisiones incorrectas, aunque con el tiempo puede ser corregible. Además se podría convertir en un racimo de datos redundantes o datos duplicados, y por consiguiente, su existencia puede provocar que la corrección no se haga en todos los sitios a la hora de realizar alguna actualización y seguramente no

se sabrá cuál de esos valores almacenado es el correcto, pues en varias ocasiones los datos son los causantes de errores de información difíciles de rastrear.

Otro de los problemas que trae consigo, se debe a que los usuarios tendrán dificultades a la hora de acceder a ciertos tipos de información y existe el riesgo añadido de que a la hora de realizar determinadas búsquedas pueda producir información errónea, siendo el peor de los resultados de un mal diseño de la base de datos. Puede repercutir negativamente en la empresa u organización propietaria de los datos. Si los datos de una base de datos van a influir en la gestión del negocio, si van a servir para decidir las actuaciones de la empresa, la base de datos debe ser una preocupación.

Aunque el diseño de una base de datos constituye un tema bastante polémico, hoy en día la información es realmente importante en la llamada “era de la información”, y en la Universidad de las Ciencias Informáticas, en el proyecto Planificación de Recursos Empresariales (ERP) no está ajeno a los problemas de diseños anteriormente planteados.

Problema a resolver

El ineficiente diseño de bases de datos para el sistema de gestión Cedrux trae consigo incoherencia entre los distintos módulos, causando problemas de rendimiento y de diseño estructural de las entidades de la base de datos.

Objeto de estudio

Sistemas de base de datos.

Campo de acción

Diseño de base de datos para el Sistema de Gestión Cedrux.

Idea a defender

Con la realización de una guía referativa para la construcción de un diseño de base de datos, se obtendrá una mayor organización y un aumento de la calidad de la base de datos del Sistema de Gestión Cedrux.

Objetivos del trabajo

General

Elaborar una propuesta de guía referativa que garantice un buen diseño de base de datos del Sistema de Gestión Cedrux.

Específicos

- Investigar los principales elementos del diseño de bases de datos.
- Modelar soluciones para el diseño de base de datos del Sistema de Gestión Cedrux.
- Validar las soluciones propuestas para el diseño de base de datos del Sistema de Gestión Cedrux.

Tareas de la investigación

- Realizar un estudio sobre los principales aspectos a tener en cuenta para obtener un buen diseño de base de datos para sistemas de gestión.
- Investigar los elementos que debe contener el diseño de base de datos de un Sistema de Gestión Empresarial para su aplicación en el diseño de la base de datos de Cedrux.
- Modelar cada uno de los elementos que pueden componer el diseño de la base de datos del Sistema de Gestión Empresarial Cedrux:
  - Normalización.
  - Tabla hash.
  - Grafos.
  - Árboles.
  - Multientidad.
  - Integración entre subsistema.
  - Entidad Atributo Valor.
  - Concurrencia.
- Validar mediante pruebas de concepto las soluciones propuestas para el diseño de la base de datos del Sistema de Gestión Cedrux.

Resultados esperados

Guía referativa donde los arquitectos de datos del proyecto de gestión ERP-Cuba, encontrarán buenas prácticas del diseño de bases de datos para el Sistema de Gestión Cedrux.

Posibles métodos de trabajo

Analítico-sintético: revisión bibliográfica profunda acerca de los diseños de base de datos.

Método empírico de la observación: para obtener información primaria acerca de diseño de base de datos y guías referativas.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza una descripción de los principales elementos teóricos que a través de la investigación realizada se consideraron importantes y que permitirá apoyar la solución del problema, permitiendo fortalecer las bases del proceso de diseño y sustentar la toma de decisiones del siguiente capítulo. Además se describen las principales herramientas que se utilizaron en la solución del problema.

1.2 Cualidades que debe presentar un diseño de base de datos

Un buen diseño de base de datos es aquel que es capaz de reflejar la estructura del problema en el mundo real, siendo capaz de representar todos los datos esperados, incluso con el paso del tiempo. Evitar el almacenamiento de información redundante, para proporcionar un acceso eficaz a la información debido a la importancia que tiene que la información sea correcta y completa, es otra de las cualidades de un buen diseño de base de datos. Si la base de datos posee información incorrecta, los reportes o los informes que recogen dicha información contendrán también información incorrecta, por lo que las decisiones que se tomen a partir de estos informes estarán mal fundamentadas. Por lo tanto, es importante que la base de datos esté bien diseñada, para asegurar su eficiencia y usabilidad a largo tiempo [2].

Son muchas las consideraciones a tomar en cuenta al momento de hacer el diseño de la base de datos, quizá las más fuertes sean [3]:

- La velocidad de acceso.
- El tamaño de la información.
- El tipo de la información.
- Facilidad de acceso a la información.
- Facilidad para extraer la información requerida.
- El comportamiento del manejador de bases de datos con cada tipo de información.

1.3 Problemas que pueden ocasionar un incorrecto diseño de bases de datos

Los problemas que pueden ocasionar un incorrecto diseño de bases de datos son [4] :

- Redundancia. Está presente cuando se repiten innecesariamente datos en los archivos que conforman la base de datos.
- Inconsistencia. Ocurre cuando existe información contradictoria o incongruente en la base de datos.
- Dificultad en el acceso a los datos. Debido a que los sistemas de procesamiento de archivos generalmente se conforman en distintos tiempos o épocas y ocasionalmente por distintos programadores, el formato de la información no es uniforme y se requiere establecer métodos de enlace y conversión para combinar datos contenidos en distintos archivos.
- Aislamiento de los datos. Se refiere a la dificultad de extender las aplicaciones que permitan controlar a la base de datos, como pueden ser, nuevos reportes, utilerías y demás debido a la diferencia de formatos en los archivos almacenados.
- Anomalías en el acceso concurrente. Ocurre cuando el sistema es multiusuario y no se establecen los controles adecuados para sincronizar los procesos que afectan a la base de datos. Comúnmente se refiere a la poca o nula efectividad de los procedimientos de bloqueo.
- Problemas de seguridad. Se presenta cuando no es posible establecer claves de acceso y resguardo en forma uniforme para todo el sistema, facilitando así el acceso a intrusos.
- Problemas de integridad. Ocurre cuando no existe a través de todo el sistema procedimientos uniformes de validación para los datos.

1.4 Proceso de diseño de la base de datos

El proceso de diseño consta de los pasos siguientes: [5]

- Determinar la finalidad de la base de datos. El principal paso para dar origen al diseño de la base de datos, es determinar la finalidad de la base de datos, para que sirva de referencia durante todo el proceso de diseño. Lo cual permitirá centrarse en los objetivos a la hora de tomar decisiones.

- Buscar y organizar la información necesaria. Se debe buscar y organizar la información necesaria, reuniendo todos los tipos de información que desee registrar en la base de datos.

- Dividir la información en tablas. Conjuntamente se pasa a dividir la información, aquellos datos que tienen relaciones como por ejemplo: en el caso de que un proveedor tenga varios productos, la información del nombre y la dirección del proveedor debe repetirse muchas veces, con lo que se malgasta el espacio en disco. Registrar la información del proveedor una sola vez

en una tabla Proveedores distinta y luego vincular esa tabla a la tabla Productos. Esta es una solución mucho mejor, pues cada tema pasará a ser una tabla.

- Convertir los elementos de información en columnas. Después se procede a convertir los elementos de información en columnas. Es decir, se decidirá qué información desea almacenar en cada tabla en el cual cada elemento se convertirá en un atributo o en un campo y se mostrará en la tabla como una columna.
- Especificar claves principales. la clave principal de cada tabla. La clave principal es una columna que se utiliza para identificar inequívocamente cada fila.
- Definir relaciones entre las tablas. De acuerdo a toda la información recopilada realizar un examen en cada una de las tablas y decidir cómo se relacionan los datos de una tabla con las demás tablas. Agregar campos a las tablas o crear nuevas tablas para especificar las relaciones según sea necesario.
- Ajustar el diseño. Realizar un análisis detallado del diseño para detectar errores. Crear tablas y agregue algunos registros con datos de ejemplo. Comprobar si puede obtener los resultados previstos de las tablas. Realizar los ajustes necesarios en el diseño.
- Aplicar las reglas de normalización: Reglas de normalización de los datos para comprobar si las tablas están estructuradas correctamente.

1.5 Fases para el diseño de un sistema de información

El diseño de una base de datos es una actividad crucial en el ambiente de datos. Una base de datos correctamente diseñada, permitirá el acceso a la información exacta y actualizada.

Para el desarrollo de un sistema de información se consta de tres fases, la primera es la fase de análisis, donde se realiza una investigación que tiene como objetivo descubrir el conjunto de requisitos de información y de procesos, que la organización necesita para cumplir sus fines. La segunda fase es la fase del diseño, la cual está compuesta por tres etapas. Más adelante se ampliará acerca de esta fase pues constituye el elemento primordial de este trabajo. Y por último, la fase de implantación, supone la incorporación del sistema de información diseñado a la organización con la puesta en marcha de la base de datos y de las aplicaciones desarrolladas.

Debido a la complejidad del proceso de diseño, conviene desglosarlo en varias etapas, en la cuales, cada unas de ellas obtiene un resultado intermedio que sirve como punto de partida a la etapa siguiente. Esto permite dividir la problemática en subproblemas, y al mismo tiempo, se simplifica el

proceso de diseño, resolviendo cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico [6].

1.5.1 Etapa del diseño conceptual

Parte de las especificaciones de requisitos del usuario, tenido como resultado una estructura de la información futura de la bases de datos, independientemente de la tecnología que se vaya a emplear. En esta etapa aún no se tienen en cuenta qué tipo de base de datos se utilizará, ni el Sistema Gestor de Base de Datos que se empleara para la manipulación de los datos, así como también tampoco se tiene en cuenta con qué lenguaje concreto se implementará la base de datos. Permitiendo al diseñador de base de datos centrarse únicamente en la problemática de la estructuración de la información sin tener que preocuparse por resolver cuestiones tecnológicas.

El resultado de la etapa del diseño conceptual se expresa mediante un modelo de datos de alto nivel. Uno de los más empleados es el modelo Entidad – Relación por Peter Chen 1976. Originalmente el modelo Entidad – Relación sólo incluía los conceptos de entidades, atributos, relación. Más tarde se añadieron otros conceptos como los atributos compuestos y las jerarquías de generalización que hoy en día se le ha denominado modelo Entidad – Relación extendido [7].

Algunas de las características generales que deben presentar todo modelo conceptual son las siguientes [1]:

- Expresividad. Suficientes conceptos para expresar perfectamente la realidad.
- Simplicidad. Deben ser simples para que los esquemas sean fáciles de entender.
- Minimalidad. Cada concepto debe tener un significado distinto.
- Formalidad. Todos los conceptos deben tener una interpretación única, precisa y bien definida.

1.5.2 Etapa del diseño lógico

Esta etapa se parte del resultado del diseño conceptual, que se transforma de manera que se adapte a la tecnología que se debe emplear. Es preciso que se ajuste al modelo del Sistema Gestor de Base de Datos con el que se desea implementar la base de datos. Por ejemplo, si se trata de un Sistema Gestor de Base de Datos Relacional, se obtendrá un conjunto de relaciones con sus atributos, claves primarias y claves foráneas [8].

Partiendo del hecho de que se ha resuelto la problemática de la estructuración de la información en un ámbito conceptual, posibilita el enfoque del diseñador en cuestiones tecnológicas relacionadas con el modelo de base de datos.

La normalización es una técnica que se utiliza en esta etapa para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura las relaciones obtenidas no tengan datos redundantes y se eliminen las anomalías de actualización.

Tanto el diseño conceptual como el diseño lógico son procesos iterativos, que tienen un punto de inicio y se van refinando continuamente. Ambas se ven como un proceso de aprendizaje, en el que el diseñador va comprendiendo el funcionamiento la empresa y el significado de los datos que maneja. Siendo las dos etapas puntos claves para conseguir que el sistema funcione correctamente.

1.5.3 Etapa del diseño físico

El diseño físico parte del esquema lógico obtenido en la fase del diseño lógico y tiene como salida la implementación de la base de datos con todas sus restricciones. La fase de diseño físico considera las estructuras de almacenamiento y los métodos de acceso necesarios para proporcionar un acceso eficiente a la base de datos en memoria secundaria. [8]

Por ello, el diseño físico depende del Sistema Gestor de Base de Datos concreto y el esquema físico se expresa mediante su lenguaje de definición de datos. Para el diseño de esta etapa es necesario haber decidido qué Sistema Gestor de Base de Datos (SGBD) se va a utilizar, pues este diseño se debe adaptar a él.

Uno de los aspectos fundamentales en esta etapa, es el aseguramiento de la seguridad de la base de datos, donde se diseñan una serie de medidas para asegurar la base de datos como por ejemplo la creación de vistas y el establecimiento de permisos para los usuarios.

1.6 Modelos de datos

Los modelos son abstracciones simplificadas de eventos y condiciones del mundo real. Por ejemplo, tales abstracciones permiten explorar las características de entidades y las relaciones entre ellas [9].

La necesidad de realizar un modelo lógicamente correcto permitirá obtener información útil y por tanto obtener buenos diseños de base de datos que son la base para buenas aplicaciones.

En el caso contrario, no se pueden esperar construir buenas aplicaciones con diseños de bases de datos incorrectamente diseñadas basados en modelos deficientes. Por lo tanto el modelo de datos lo

que intenta hacer es una reproducción de una información real que se desea almacenar en un sistema informático. Siendo su principal objetivo ayudar a modelar la situación de un problema.

1.6.1 Características del modelado.

Las razones por las cuales los diseñadores de base de datos emplean el modelado como una herramienta de comunicación, es debido a su simplicidad y para facilitar la interacción entre el diseñador, programador de aplicación y el cliente. Unas de las características que presenta el modelo de datos es su independencia del gestor de base de datos que se vaya a utilizar, no importa si se trabaja con MySQL, PosgreSQL o cualquier otro gestor de base de datos. El modelado no está orientado a ningún gestor específico.

En el modelado de datos sólo se debe reflejar la existencia de los datos más relevantes para el sistema, no importa su dominio, ni que se hará con los datos, tampoco se tiene en cuenta las restricciones de espacio, almacenamiento o tiempo de ejecución.

Un modelo de datos para una base de datos no es más que una colección de conceptos que se emplean para describir la estructura de una base de datos. Esa colección de conceptos puede incluir entidades atributos y relaciones [10].

1.6.2 Tipos de modelos de datos

Los modelos de datos pueden agruparse en dos categorías principalmente: los modelos conceptuales y los modelos de ejecución [9].

- El modelo conceptual. Este modelo se enfoca hacia lo que está representado la base de datos y no cómo está representado. Incluye el modelo Entidad - Relación y el modelo orientado objeto.
- El modelo de ejecución. En contraste al modelo conceptual este modelo si hace énfasis en como los datos están representados en la base de datos o en cómo se ejecuta la estructura de datos para representar lo que esta modelado. Los modelos de ejecución incluyen modelos de base de datos jerárquicos, el modelo de base de dato de red, y el modelo de base de datos relacional, que en este caso es el que se utilizará en el sistema de gestión empresarial Cedrux.

1.6.2.1 Modelos de base de datos jerárquico

El modelo jerárquico fue desarrollado para permitir la representación de aquellas situaciones de la vida real en las que predominan las relaciones de tipo 1: N, y también las relaciones de tipo 1:1 [11].

La manera en que se representa la información es mediante una estricta relación de padre e hijo, en el cual se organizan por niveles en dependencia de su jerarquía, de manera que un padre puede tener más de un hijo, todos ellos localizados en un mismo nivel y un hijo tiene únicamente un padre situado en el nivel inmediatamente superior al suyo. Debido a esta estricta relación se presentan una de sus principales desventajas, un ejemplo de esto es en el caso en que se tenga que darle de baja a un padre, por lo que necesariamente se tendría que darle baja a todos sus hijos, ya sean o no descendientes directos.

Las entidades se denominan en el caso particular del modelo jerárquico segmentos, mientras que los atributos reciben el nombre de campos. Los segmentos, se organizan en niveles de manera que en un mismo nivel estén todos aquellos segmentos que dependen de un segmento de nivel inmediatamente superior.

Una base de datos de este tipo, no permite el acceso directo a las instancias de un segmento hijo, si no es seleccionando previamente las instancias de los padres de los que depende. Siendo otros de sus problemas.

1.6.2.2 Modelo de datos de red

Este es un modelo ligeramente distinto del jerárquico; se diferencia fundamental en la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). Fue una gran mejora respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales [11].

1.6.2.3 Modelo de datos relacional

Uno de los puntos fuertes en el modelo relacional es la sencillez de su estructura lógica donde todos sus datos están estructurados en formas de tablas formadas por columnas y filas. Es modelo el más utilizado debido a su rápido entendimiento por parte de los usuarios que no tienen conocimientos profundos sobre diseño de bases de datos. [12]

Codd propuso que los sistemas de bases de datos deberían presentarse a los usuarios con una visión de los datos organizados en estructuras llamadas relaciones, definidas como conjuntos de tuplas (filas) y no como series o secuencias de objetos, con lo que el orden no es importante. Realiza énfasis en que el usuario de un sistema relacional sólo debía preocuparse por el qué consultar y no el cómo de las estructuras de almacenamiento (lo que ahora se conoce como modelo físico) [13].

Las bases de datos relacionales pasan por un proceso que permitirá que la base de datos sea utilizada de manera óptima, este proceso se le conoce como normalización de la base de datos.

Unas de las características que ofrece este modelo relacional es que los usuarios no necesitan saber donde se encuentran los datos físicamente, y permite ampliar el esquema conceptual sin las modificaciones de las aplicaciones de gestión. Además que puede ser entendido y usado por cualquier usuario.

El hecho de que las bases de datos relacionales estén basadas en dos ramas de la teoría de las matemáticas, la teoría de conjuntos y la lógica de predicado es la ventaja que la hace robusta, segura, fiable y predecible. Un ejemplo a considerar sería, si se tiene dos tablas que están relacionadas se podrá extraer datos de las dos tablas a la vez, simplemente por el modo en que funciona la teoría en la base de datos relacionales. Los datos que se saquen de ambas tablas tendrán valores coincidentes del campo que ambas tablas tienen en común.

1.7 Guías de diseño

En la actualidad, tanto en el ámbito nacional como internacional existen guías para el diseño exitoso de una base de datos, con el objetivo de conducir, dirigir o encaminar a cualquier persona interesada en la construcción de un sistema de información, en especial a los arquitectos de datos.

Ejemplos de guías de diseños:

1. Diseño de base de datos escrito por Rafael Camps, Luis Albert Casillas, Dolors Costal, Marc Gilbert, Carme Matín y Oscar Pérez. La guía fue realizada en el año 2005.

   Constitución de la guía: Está compuesta por ocho módulos didácticos entre los que se encuentra el módulo de introducción al diseño de base de datos con el objetivo de aprender a modelar y representar gráficamente una base de datos, así como también detectar los posibles problemas de diseño antes que estos puedan afectar a la aplicación y a construir bases de datos eficientes para los distintos casos de relaciones que formaran la base de datos. Los módulos restantes realizan una descripción de aspectos tales como la arquitectura del sistema gestor de base de datos, características, acceso al servidor, creación y manipulación de los datos especializado en dos de los gestores de base de datos usados en el ámbito del software libre como son PostgreSQL y MySQL. También se realiza una descripción de los aspectos más importantes del modelo relacional donde se plasman una serie de ventajas que conlleva la utilización de este.

2. Guía de aprendizaje: Diseño conceptual del diseño de base de datos escrito por Jorge Sánchez en el año 2004, profesor de informática de Palencia.

   Constitución de la guía: En esta se realiza una descripción de los elementos que componen la etapa del diseño conceptual de la fase de diseño para la construcción de un sistema de información. En ella se tratan fundamentalmente los conceptos básicos para la construcción de un Modelo Entidad – Relación, así como la clasificación de los modelos de datos y principales ventajas y desventajas de una base de datos.

3. Diseño de base de datos proporcionada por la Universidad de Sevilla.

   Constitución de la guía: En el material se realiza una descripción del ciclo de vida de un sistema de información, donde se examinan las entradas y salidas de cada una de las fases implicada en el desarrollo de este. Se detallan conceptos correspondientes a cada fase, el objetivo de estas así como las características y la necesidad de guardar sólo la información imprescindible.

De manera general, la mayoría de estas guías abarcan principalmente en su contenido los conceptos básicos y terminologías fundamentales, así como la descomposición del proceso de diseño en fases y los principales objetivos de cada una de estas, además de las técnicas para conseguir estos objetivos.

Mediante la descripción de los ejemplos seleccionados, se puede observar que ellas carecen de elementos que caracterizan a un ERP: como son multientidad, multimoneda, multiusuario, concurrencia, integración y otras, que harán de la base de datos una mayor organización y que responda de forma eficaz a los requerimientos planteados por los usuarios finales.

Dentro de los proyectos existentes en la Universidad de la Ciencias Informáticas, se encuentra ERP – Cuba, el cual ha encaminado su diseño basándose en manuales y documentos específicos debido a la inexistencia de una guía que contenga patrones que describan cómo realizar cada una de las características específicas que debe poseer una base de datos para un Sistema ERP.

Por tales razones en este trabajo se propone una guía que contengan los elementos más relevantes de las guías estudiadas, además de las soluciones que fueron definidas por la Subdirección de Tecnología como imprescindibles y que responden a características de los sistemas ERP.

1.8 Diseño en el ERP- Cuba

¿Qué es un ERP?

Los sistemas de Planificación de Recursos Empresariales (ERP según sus siglas en inglés): son sistemas integrales de gestión para las empresas. Se caracterizan por estar compuestos por diferentes

partes integradas en una única aplicación. Estas partes son de diferente uso, por ejemplo: producción, ventas, compras, logística, contabilidad (de varios tipos), gestión de proyectos, GIS (sistema de información geográfica), inventarios y control de almacenes, pedidos, nóminas, entre otros. Se podrá definir un ERP como la integración de todas estas partes [15].

La propia definición de ERP indica la necesidad de "Disponibilidad de toda la información para todo el mundo todo el tiempo".

Otras características destacables de los sistemas ERP son [15]:

- Los componentes del ERP interactúan entre sí consolidando todas las operaciones.
- En un sistema ERP los datos se ingresan sólo una vez y deben ser consistentes, completos y comunes.

Por decisión de la Subdirección de Tecnología se decidió adoptar el modelo relacional como el modelo de base de datos a utilizar por el Proyecto ERP-Cuba para el diseño de la base de datos del Sistema integral de Gestión Cedrux, teniendo en cuenta que el Sistema Gestor en el cual se desarrolla la base de datos es PostgreSQL, y el mismo está basado en este modelo.

A continuación una serie de consideraciones a tener en cuenta:

El Dr. Codd, hizo la siguiente reflexión.

"…El contenido entero de una base de datos relacional se representa por una y solo una forma, a saber: como valores de atributos en tuplas dentro de relaciones." [13]

¡Sabias palabras!

En el modelo relacional de bases de datos, no importa el lugar y la forma en que se almacenen los datos (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un nuevo usuario de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para gestionar la información. El modelo relacional es un estándar de la industria consolidado, una tecnología confiable y eficiente que estará por muchos años más antes de que sea desplazada por una nueva y mejor [13].

Se puede decir que su enorme éxito no se debe a que permite de forma implícita operaciones conceptualmente abstractas sobre los datos, sino que es una solución simple y elegante para satisfacer las más diversas condiciones de consulta y extracción de datos e información y que posee

altos niveles de fiabilidad e integridad en el manejo de grandes cantidades de datos, razón por la cual es en el que se basan la mayoría de los Sistemas Gestores de Base de Datos comerciales en uso hoy en día, para la construcción de bases de datos y así mismo para las grandes aplicaciones de gestión, entre ellas los proyecto de Planificación de Recursos Empresariales (ERP).

Haciendo uso de este modelo relacional, en sistema integral de gestión Cedrux se han desarrollado un grupo de soluciones para dar cumplimiento a los requerimientos del sistema y para el aseguramiento del buen funcionamiento del mismo, aprovechando las ventajas que brinda el modelo relacional de bases de datos.

Dentro de las soluciones que se han realizado haciendo uso del modelo relacional se encuentran:

- La multientidad que tiene como objetivo que en una misma base de datos convivan los datos de más de una entidad, permitiendo el compartimiento de tablas comunes entre varias entidades que hacen uso de la multientidad.
- La concurrencia, con el objetivo de en caso en que dos o más usuario se conecten a un mismo registro de una tabla en la base de datos para realizar alguna modificación, no se produzcan inconsistencias en los datos.
- La utilización de tipo de datos abstracto árbol en varios módulos del proyecto que requiera su uso por las ventajas que trae, como por ejemplo el módulo de contabilidad en el nomenclador de cuenta, ya que una cuenta puede tener varias subcuentas y así sucesivamente, estableciendo una jerarquía.

Una de las decisiones que se han tomado en la Subdirección de Tecnología es la creación de un módulo para concentrar todos los nomencladores de la base de datos y simplemente hacer uso de ellos con el objetivo de eliminar la repetición de tablas en varios esquemas que pueden estar centralizadas en uno sólo y ser usadas mediante la integración con este esquema.

Debido a que en el proyecto han ocurrido problemas en el diseño de la base de datos, causando problemas de rendimiento, ya sea por que los arquitectos de datos de cada módulo han realizado el diseño de la base de datos como estimó conveniente, o bien porque se realizó en la marcha del desarrollo o porque no existía un documento rector que plasmara un grupo de soluciones de buenas prácticas para proyectos de gestión empresarial. Sin la existencia de este documento los arquitectos carecían de una guía para realizar su trabajo, lo que posibilitó la pérdida de tiempo e incoherencia entre los distintos módulo que lo componen.

Por tales motivos, la Subdirección de Tecnología y el arquitecto de dato principal identificaron un conjunto de soluciones y buenas prácticas de diseño que permitirán eliminar gran parte de los problemas que hoy acosan en la base de dato del Sistema de Gestión Empresarial Cedrux.

1.8.1 Buenas prácticas y eficiencia para el diseño

1.8.1.2 Interrelaciones entre tablas

Las tablas son una estructura bidimensional compuesta por filas y columnas, cada fila de la tabla representa la ocurrencia de una sola entidad dentro de un conjunto de entidades, cada columna representa un atributo y tiene nombres distintos. Donde cada tabla está compuesta por una serie de atributos que identifica de manera única a cada fila.

Las interrelaciones son asociaciones o conexiones que existen entre dos o más entidades. Las entidades relacionadas pueden pertenecer al mismo o a distintos conjuntos de entidades. Siendo la relación el pilar fundamental en el diseño general de un sistema de base de datos relacional, permitiendo acceder a la información almacenada en varias tablas mediante esta. Además estas asociaciones permiten facilitar el entendimiento del negocio mediante el diagrama Entidad – Relación construido en la fase del diseño conceptual [16].

Existen dos formas de clasificar a una relación entre dos tablas, clasificación por modalidad y clasificación por cardinalidad [17].

1.8.1.2.1 Clasificación por modalidad:

- Obligatoria. Dadas las tablas A y B, que se encuentran relacionadas, si para todo registro de A debe existir siempre al menos un registro de B, se dice que la relación en sentido A->B es Obligatoria.
- Optativa. Dadas las tablas A y B, que se encuentran relacionadas: Si para todo registro de A, pueden existir o no, uno o varios registros de B asociados, se dice que la relación en sentido A->B es Optativa. La modalidad de las relaciones se debe analizar en ambos sentidos.

1.8.1.2.2 Clasificación por Cardinalidad:

- Relación Uno a Uno. Una entidad A se asocia a lo sumo una entidad en B, y una entidad en B se asocia a lo sumo una entidad en A. Donde la llave primaria de cualquiera de las dos entidades va como llave foránea para cualquiera de las dos entidades.

- Relación Uno a Muchos. Una entidad en A se asocia con cualquier número de entidades en B. Una entidad en B, sin embargo, se puede asociar con a lo sumo una entidad en A. Donde la llave primaria va como llave foránea para la entidad de mayor cardinalidad de muchos.
- Relación Muchos a Muchos. Una entidad en A se asocia con cualquier número de entidades en B, y una entidad en B se asocia con cualquier número de entidades en A. Donde se crea una nueva tabla AB en la que toma como llave primaria la combinación de las llaves primarias de las entidades asociadas por la interrelación.

1.8.1.3 Normalización

El diseño conceptual tiene como salida un modelo Entidad – Relación donde se trata de obtener todos los datos necesarios para poder soportar todas las funciones a las que servirá el sistema final, identificando las entidades y sus relaciones. Sin embargo, estas tablas han sido obtenidas a partir del diseño conceptual, elaborado sin ningún tipo de reglas, lo que pudiera traer consigo una serie de problemas tales como: redundancia de datos y anomalías de actualización.

La existencia de estados no válidos en la base de datos, es lo que se conoce como anomalías de actualización que pueden ser de tres tipos[18]:

- Inserción: imposibilidad de adicionar datos en la base de datos debido a la ausencia de otros datos.
- Borrado: pérdidas no intencionadas de datos debido a que se han borrado otros datos.
- Modificación: inconsistencias de los datos como resultado de datos redundantes y actualizaciones parciales.

Por tales motivos, el proceso de normalización tiene como objetivo producir un conjunto de relaciones con una serie de propiedades deseables, eliminando los problemas anteriormente mencionados.

Este proceso comienza examinando directamente los atributos y las relaciones que existen entre ellos. A partir de ahí se utilizan las formas normales que son una serie de pruebas por la que debe pasar la relación para tratar de identificar el agrupamiento óptimo de los atributos. Permitiendo a los diseñadores de la base de datos desarrollar un esquema que minimice los problemas de lógica, y cada unas de estas reglas está basada en la que le antecede.

Formas Normales[19]:

- Primera Forma Normal (1FN). Una relación está en primera forma normal si y sólo si los dominios de sus atributos son atómicos.

- Segunda Forma Normal (2FN). Una relación está en 2FN si está en 1FN y todo atributo que no sea clave depende funcionalmente, de manera completa de la clave.
- Tercera Forma Normal (3FN). Una relación está en 3FN si está en 2FN y no existen dependencias transitivas entre los atributos no claves, con respecto a la clave
- Forma Normal de Boyce-Codd (FNBC). Una tabla está en FNBC si está en 3FN y todos sus atributos son llaves candidatas o llaves primaria.

Existe también la Cuarta y la Quinta Forma Normal pero son las menos utilizadas, este documento abarcará hasta las FNBC tomando como punto de partida que muchos autores consideran que los diseñadores de base de datos no tienen que normalizar hasta la forma normal más alta posible. Las relaciones pueden dejarse en formas normales inferiores por razones de rendimiento. Además de que a través del desarrollo de la base de datos de Cedrux la experiencia ha mostrado que normalizando las tablas hasta la 3FN y en algunos casos hasta la FNBC, el diseño es eficiente y satisface la mayoría de las necesidades de los diseñadores.

Antes de normalizar un diseño es importante que se tenga en cuenta los tipos de las relaciones que pueden existir entre las entidades de la base de datos y de esta manera se podrá normalizar con mayor facilidad y comprensión de hacia dónde puede y debe llevarse el diseño de una solución determinada.

Codd plantea que: “Ni la 1FN ni la 2FN se consideran buenos diseños de esquemas de relación, constituyen solo escalones para llegar a la 3FN y FNBC”[20] .

1.8.1.4 Generalización/ Especialización

La generalización - especialización es un principio de herencia, en las cuales las entidades de bajo nivel heredan todos los atributos de las entidades de mayor nivel. Es importante tener en cuenta que las entidades de bajo nivel no tienen llave primaria, únicamente la entidad de mayor nivel es la que tiene entre sus atributos la llave y que como consecuencia las heredan las del nivel inferior. Si se tiene en cuenta el sentido de la entidad del nivel superior hacia las entidades de nivel inferior estaría presente la especialización, debido a que cada entidad que heredan de ella se especializa en funciones específicas de su propio entorno. En caso contrario, si el sentido fuera desde las entidades inferiores hasta la entidad superior, estaría presente la generalización.

Según Víctor García Gózales la generalización/ especialización consiste en identificar en las entidades, aquellos atributos semejantes para formar una nueva entidad general, la cual dicha entidad

quedará ubicada en un nivel superior que las entidades especializadas, con el objetivo de tratar de eliminar la redundancia o la repetición de los atributos[21].

En ocasiones se tienen un conjuntos de determinas entidades, en las cuales, entre ellas existe similitud, en el sentido de tener varios atributos en común entre ellas. Por lo tanto, esta similitud entre estas entidades se puede expresar aplicando el principio de generalización, que es una relación contenedora que existe entre el conjunto de entidades de nivel más alto y uno o más conjuntos de entidades de nivel más bajo. Siendo una propiedad crucial entre las entidades del nivel superior y las del nivel inferior la herencia de atributos, donde los atributos de la entidad del nivel superior son heredados por el conjunto de entidades del nivel inferior.

Para modelar soluciones donde se vaya a usar la generalización/ especialización, el diseñador de la base de datos puede elegir colocar ciertas restricciones sobre las generalizaciones para controlar la forma en que se va a comportar el diseño. Un tipo de restricción implica determinar que entidades pueden ser miembros de un conjunto de entidades de nivel más bajo dado.

Tales relaciones de miembros pueden ser algunos de los siguientes[21]:

- Definido por condición. En los conjuntos de entidades de nivel más bajo, la relación miembro se evalúa en función de si una entidad satisface o no una condición explícita o predicado. Si todas las entidades de nivel más bajo se evalúan en función del mismo atributo, este tipo de generalización se denomina definido por atributo.
- Definido por el usuario. Los conjuntos de entidades de nivel más bajo definidos por el usuario no están restringidos mediante una condición de miembro; en cambio, las entidades se asignan a un conjunto de entidades dado por el usuario de la base de datos. En este caso la entidad generalizada no se asigna a una entidad especializada automáticamente en términos de una condición que lo defina explícitamente, sino que es el usuario a cargo de esta decisión quien realiza la acción.

Otros tipos de restricciones se definen según si las entidades pueden pertenecer a más de un conjunto de entidades de nivel más bajo en una generalización simple. Los conjuntos de entidades de nivel más bajo pueden ser uno de los siguientes[19]:

- Disjunto. Una restricción sobre el carácter disjunto requiere que una entidad no pertenezca a más de un conjunto de entidades de nivel más bajo. Es decir, que una entidad cuenta puede satisfacer sólo una condición para un atributo de manera que dicha entidad puede ser bien una u otra de las entidades especializadas, pero no más de una a la vez.

- Solapado. En las generalizaciones solapadas, la misma entidad puede pertenecer a más de un conjunto de entidades de nivel más bajo en una generalización simple.

Una restricción final es la restricción de completitud en una generalización o especialización, específica si un conjunto de entidades de nivel más alto debe pertenecer o no al menos, a uno de los conjuntos de las entidades de nivel más bajo en una generalización/especialización.

Esta restricción puede ser una de las siguientes[19]:

- Generalización o especialización total. Cada entidad de nivel más alto debe pertenecer a un conjunto de entidades de nivel más bajo.
- Generalización o especialización parcial. Algunas entidades de nivel más alto pueden no pertenecer a algún conjunto de entidades de nivel más bajo.

1.8.2 Herramientas

Para la realización de las soluciones propuestas en el siguiente capítulo se utilizan tres herramientas siguiendo el marco de trabajo propuesto por el proyecto, el cual define las tecnologías y herramientas a utilizar. A continuación se ofrece una panorámica de las mismas.

1.8.2.1 Herramienta de diseño: Visual Paradimg

Herramienta Case (Computer Aided Software Engineering), que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue[22].

Para la creación, la modificación y las relaciones de las tablas, debe utilizarse siempre una herramienta capaz de mantener los objetos físicos en relación con el modelo lógico, de manera que se logre la documentación. A fin de lograr este propósito, la herramienta a utilizar para la creación y sincronización hacia o desde la base de datos, es el Visual Paradigm, ya que permite el trabajo con PostgreSQL.

Se posee una licencia para su uso (adquirida por la universidad) y funciona en las dos plataformas utilizadas en el desarrollo (Linux, Windows).

1.8.2.2 Herramienta de implementación:

EMS PostgreSQL Manager: Es una herramienta de gran alcance para la administración y el desarrollo del servidor de la base de datos de PostgreSQL, ya que ofrece varias herramientas, que complementan diversas funcionalidades, que dan gran alcance a los usuarios[23].

PostgreSQL: Es un sistema de gestión de bases de datos objeto-relacional (ORDBMS), el cual incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones,

restricciones, disparadores, reglas e integridad transaccional, este ORDBMS está basado en el proyecto POSTGRES, de la universidad de Berkeley. PostgreSQL es un poderoso manejador de base de datos de código abierto liberado bajo la licencia BSD, diseñado para administrar grandes volúmenes de datos. Se ejecuta en varios Sistemas Operativo como Linux, Windows entre otros.

EMS Data Generator for PostgreSQL: Es una impresionante herramienta para la generación de datos de prueba para PostgreSQL en las tablas de la base de datos con la posibilidad de guardar y editar secuencias de comandos. Ayuda a simular el entorno de producción de la base de datos y le permite rellenar varias tablas de la base de datos en PostgreSQL con datos de las pruebas.

1.8.3 Elementos teóricos de las soluciones para el diseño de un Sistema ERP

1.8.3.1 Tabla Hash

Una tabla hash es una estructura de datos el cual asocia llaves o claves con valores, con el objetivo de realizar búsquedas muy rápidas. Una tabla hash permite un almacenamiento y una posterior recuperación eficiente de elementos o valores a partir de un objeto, denominado clave, en donde cada uno de los valores o datos a insertar se le asigna mediante una fórmula matemática (función hash), una posición única en la tabla, almacenando la información a partir de la clave generada en función de la función hash, con lo que su búsqueda, inserción y eliminación son de tiempo de ejecución O (1)[24]. La principal desventaja de esta estructura de datos es cuando se tienen datos duplicados, por lo que se produce colisiones que no es más que dos palabras que se le asigna una misma posición. Problema que se resuelve mediante una función de resolución de colisiones explicada en el capítulo dos.

1.8.3.2 Grafos

Un grafo es un tipo de dato abstracto (TDA) no lineal, compuesto por un conjunto finito de vértices denominados nodos, y un conjunto de relaciones finitas entre dos o más nodos llamadas aristas. La representación de esta estructura de datos se puede ver representada en la cotidianidad, como pueden ser las carreteras, líneas telefónicas, líneas de televisión por cable, el transporte colectivo, metro, circuitos eléctricos de nuestras casas, automóviles, y otras cosas más.

La utilización de un grafo en una base de datos, trae una serie de ventajas como por ejemplo flexibilidad en el cambio de estructura, unificación en la representación de los datos, esquemas y consultas, se puede recorrer directamente la base de datos de forma jerárquica, obtener el nodo abuelo del nodo y viceversa. Las bases de datos orientadas a grafos estuvieron siendo utilizadas por un tiempo, sin embargo dada la aparición de las bases de datos relacionales, las bases de datos

orientada a grafo pasaron a un segundo plano, debido principalmente, por la simplicidad y fácil manejo del modelo relacional.

EL uso de la bases de datos orientadas a grafos es escaso, y actualmente hay muy pocas herramientas para su desarrollo, una de las más conocida es la “Plataforma G”, la cual es un gestor de bases de datos orientada a grafos desarrollado por el ingeniero Alfonso Ríos Alonso[25].

1.8.3.3 Árboles

Existen diversos problemas cuya solución no puede representarse mediante una disposición secuencial de elementos, por ejemplo, análisis de circuitos eléctricos, representación de fórmulas matemáticas, la estructura sintáctica de un programa fuente en los compiladores, el índice de un libro, etc. Estos problemas se resuelven utilizando estructuras de datos no lineales, las cuales representan los datos en una manera mucho más cercana a la realidad.

Una de las estructuras de datos no lineales más utilizadas es el árbol. La estructura de árbol expresa una relación jerárquica entre los elementos de un conjunto dado.

En algunos casos la estructura del tipo de dato abstracto árbol es la más adecuada para la organización de la información, siendo además muy eficiente en cuanto a tiempo computacional en algunas operaciones, algunos ejemplos de ello son[26]:

- Búsqueda y ordenamiento.
- Codificación de la información.
- Análisis de la sintaxis de los lenguajes de programación.
- Representación del conocimiento y búsqueda de soluciones.

1.8.3.4 Multientidad

Más de uno se han encontrado con la problemática de que en una misma base de datos, convivan los datos de más de una entidad e incluso entre las mismas se utilicen o se comparta los datos en algún momento o proceso específico de una determinada entidad, siempre y cuando cada una de ellas tenga acceso a los datos en la medida que se haya definido y con los permisos que se hayan configurado para el acceso a la misma.

Muchas han sido las soluciones propuestas para esta problemática por más de una organización que se ha encontrado con la necesidad de resolver dicho problema. Este fenómeno se conoce con el nombre de Multientidad o Multiempresa, y no es más que la convivencia en una misma base de datos

de la información de varias empresas, de manera tal que cada una de ellas sólo tenga acceso a la información que le corresponde sin tener que violar, ni alterar la información restante.

1.8.3.5 Integración

La palabra integración tiene su origen en el concepto latino integratïo. Se trata de la acción y efecto de integrar o integrarse (constituir un todo, completar un todo con las partes que faltaban o hacer que alguien o algo pase a formar parte de un todo)[27].

Al inicio del epígrafe 1.7 se mencionó que un ERP, se puede definir como la integración de todas estas partes, siendo la diferencia fundamental entre un ERP y otra aplicación de gestión. El ERP integra todo lo necesario para el funcionamiento de los procesos de negocio de la empresa. No se puede hablar de ERP en el momento que tan sólo se integra uno o una pequeña parte de los procesos de negocio. La propia definición de ERP indica la necesidad de "Disponibilidad de toda la información para todo el mundo todo el tiempo".

La integración de toda esa información en una base de datos, permite la optimización de los procesos y la obtención de la información de manera más rápida y precisa y que todos los usuarios puedan compartir la información y acceder a ella en forma constante.

Una de las características fundamentales que diferencian al ERP de otro software de gestión es la integridad de sus sistemas. Pero además cuenta con más particularidades que influyen en su diferenciación con otras aplicaciones, y que son la división interna en módulos, lo que permite que se vayan instalando según las necesidades de cada cliente y con ello la adaptabilidad.

Por decisión de la Subdirección de Tecnología del Proyecto ERP-Cuba se debe crear un esquema por cada módulo que forme parte de Cedrux y luego se debe garantizar la integración de estos esquemas. De esta forma funciona como un todo como describe la definición de un ERP.

Es importante que se tenga en cuenta que las bases de datos de los sistemas ERP, no es sólo integrar varios departamentos de una empresa. Para verdaderamente ser considerado ERP, el sistema debe poseer algunas de las siguientes características fundamentales[28]:

- Flexibilidad: un sistema ERP es flexible si responde a las constantes transformaciones de las empresas.
- Modular: el sistema ERP, es un sistema de arquitectura abierta, puede usar un módulo libremente sin que este afecte los restantes.

- Escalabilidad: es la propiedad deseable de un sistema, una red o un proceso, que indica su habilidad para, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.
- Integridad de los datos: el término integridad de datos se refiere a la corrección y completitud de los datos en una base de datos.
    - Integridad referencial: asegura la integridad entre las claves ajenas y primarias. Debe asegurarse que no ocurra ninguna actualización que pueda corromper la integridad referencial.
- Seguridad y protección de los datos
    - Protección: garantizar el acceso autorizado a los datos, de forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.
    - Seguridad: el sistema de bases de datos debe disponer de métodos que garanticen la restauración de las bases de datos al producirse alguna falla técnica, interrupción de la energía eléctrica, etc.

Esta es una característica que es responsabilidad del gestor pero la base de datos debe permitir que se lleve a cabo este tipo de cuestiones.

1.8.3.6 Entidad Atributo Valor

EL modelo Entidad Atributo Valor (EAV), se usa en los casos donde número de atributos usados para describir una entidad es muy grande, pero que aplicados individualmente a una entidad concreta es pequeño[29].

Este modelo se puede aplicar cuando existen las siguientes características:

- Un gran conjunto de atributos, sólo algunos de los cuales se aplicará a una entidad individual.
- Un gran conjunto de atributos que tiende a cambiar con frecuencia en el tiempo.
- Atributos que pueden tener valores muy largos.

Los problemas que aporta este modelo son:

- Consultas complejas, que en algunos casos deben usar herramientas típicas de datawarehouse, para convertir en columnas lo que, en realidad, son filas.
- Tipo de dato único para todos los atributos, sean numéricos, fechas o cadenas, que luego deberán convertirse, o complicar el modelo con distintas tablas de atributos en función del tipo de dato.

Para resolver los atributos que tienden a cambiar con frecuencia en el tiempo, se recurre a la estructura de dato EAV. Este modelo puede hacer que sea mucho más fácil de añadir, eliminar y modificar una amplia gama de atributos de la entidad gestionada por una aplicación web.

1.8.3.7 Control de concurrencia

Generalmente las bases de datos se utilizan en entorno multiusuario, en la cual varias personas desde una determinada aplicación pueden conectarse a la misma vez a una misma base de datos. Este acceso paralelo en caso de que se realizara alguna actualización de la información en la base de datos puede dar como resultados informaciones inconsistentes o incorrecta.

La concurrencia no es más que la capacidad de un sistema de manejar modificaciones realizadas simultáneamente sobre una base de datos, siempre y cuando cada una de las transacciones realizadas se realice sin la violación de la integridad de los datos de la base de datos, donde cada transacción debe ser ejecutada con seguridad y siguiendo las reglas ACID (*Atomicidad, consistencia, aislamiento, durabilidad*) características que garantizará que las transacciones en la base de datos se procedan confiablemente[30]:

- Atomicidad. Es la propiedad que asegura que la operación en la base de datos sea realizada o no, y antes de algún fallo en el sistema, no puede quedar a medias la operación.
- Consistencia. Característica que garantizará que solo se empiezan aquellas operaciones que se puedan terminar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de la integridad de la base de datos.
- Aislamiento. Propiedad que asegura que la realización de dos transacciones sobre la misma información sean independientes y que no se generen ningún tipo de error, esta característica asegura que una operación no pueda afectar a otra.
- Durabilidad. Propiedad que asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer aunque falle el sistema.

El punto importante aquí es asegurar que la base de datos regrese a un estado consistente al terminar la ejecución de una transacción.

CONCLUSIONES

En este capítulo se abordaron los elementos teóricos que sustentan la solución del problema. Partiendo de la necesidad de contar con una guía de diseño de base de datos para los arquitectos de datos para cualquier Sistema de Gestión en especial Cedrux, se analizaron los principales elementos

del diseño de base de datos, permitiendo conocer las ventajas y los inconvenientes que puede ocasionar si se realiza un mal diseño.

Un criterio importante al que se arribó, es que el diseño de la base de datos es un proceso compuesto por tres etapas en las cuales cada una de ellas obtiene un resultado intermedio que sirve como punto de partida de la etapa siguiente, por lo que si se sigue detalladamente cada de los pasos que tiene cada una de estas tres fases, se simplifica el proceso de diseño resolviendo cada uno de los objetivos trazados en cada fase utilizando técnicas específicas para cada una de ellas.

También permitió fortificar la base para el proceso de diseño y sustentar la toma de decisiones del capítulo siguiente. Las herramientas utilizadas permitirán diseñar y validar las soluciones que se abordarán en los capítulos posteriores entre ellas se encuentra PostgreSQL definido por la Subdirección de Tecnología como el Sistema Gestor de Base de Datos a utilizar para Cedrux basado en el modelo relacional y una vez descritas las ventajas que este trae, las soluciones que en el siguiente capítulo se proponen estarán basadas en este modelo.

CAPÍTULO 2: GUÍA REFERATIVA. DESCRIPCIÓN DE LA SOLUCIÓN

2 Introducción

En este capítulo se realiza una propuesta de soluciones que constituyen buenas prácticas en el diseño de una base de datos para proyectos de gestión, permitiendo a los arquitectos de datos del Sistema Gestión Cedrux, poseer una mayor organización en la misma.

2.1 Buenas prácticas de diseño

2.1.1 Interrelaciones entre Tablas

2.1.1.1 Relaciones de uno a uno

Suponga que en la universidad se quiere realizar un diseño que permita almacenar los decanos que tienen asignado o no asignatura. De cada decano se conocen el nombre y el salario, y de las asignaturas solo se conoce el nombre o la denominación y su abreviatura. Es importante aclarar que en este caso, un decano puede impartir sólo una asignatura y una asignatura, es impartida por sólo un decano. Y un decano puede no tener asignada ninguna asignatura a impartir, pero cada asignatura existente tiene asignado el decano que la impartirá.

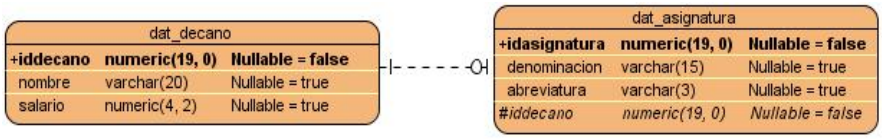

Figura 2.1. Ejemplo de relación de uno a uno

Descripción del diseño:

El diseño está conformado por dos tablas:

Tabla 1: dat_decano que contiene tres columnas: iddecano: identifica a los decanos; nombre: guarda el nombre de los decanos; salario: registra el salario que reciben los mismos por su labor.

Tabla 2: dat_asignatura, compuesta por cuatro campos: idasignatura: identifica cada asignatura a impartir; denominacion: guarda el nombre de las asignaturas; abreviatura: guarda la abreviatura de la denominación de las asignaturas; iddecano: guarda el decano que la imparte.

Es importante especificar hacia que tabla irá la llave primaria, pero eso depende de lo que requiera el problema. En el caso anterior interesaba conocer a que decano pertenecían las asignaturas guardadas en la tabla dat_asignatura, por lo que iddecano pasa como llave foránea para la tabla dat_asignatura.

2.1.1.2 Relaciones de uno a muchos

En la relación de uno a muchos, puede estar indefinida la cantidad de elementos que se pueden almacenar en la tabla que corresponde a la parte de los muchos, pero en muchas ocasiones esto es restringido a una determinada cifra por cuestiones de necesidades en el problema que se diseña. Si relación de uno a muchos tuviera restricciones en la parte del mucho, la situación cambiaría en cuanto a su implementación, por lo que en este caso, se validaría que no se inserte más de la cantidad definidas de los elementos establecidos por la restricción. Un ejemplo de esto, es en el caso de la relación de uno a muchos sin restricciones, donde un curso sólo podrá aceptar una matrícula de 30 estudiantes, lo cual pudiera tener sentido para alguna organización o institución educacional y en ese caso se debe validar mediante una función de inserción que la cantidad máxima de alumnos por curso es de 30.

A continuación se presenta la manera en que se puede diseñar las relación de uno a muchos en una base de datos, en este caso no tiene definida la cantidad de alumnos que puede existir en cada curso. Es importante aclarar que en este diseño, un curso tiene muchos alumnos y un alumno está en un único curso, y un curso puede estar transitoriamente vacío y todo alumno tiene un curso asignado.

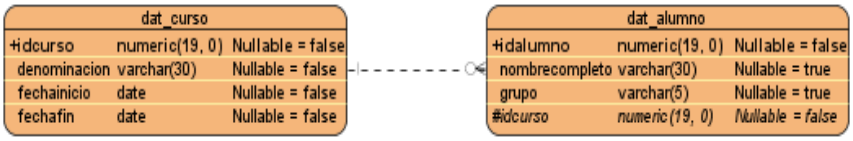

Figura 2.2. Ejemplo de relación de uno a muchos.

Descripción del diseño:

El diseño está conformado por dos tablas:

Tabla 1: dat_curso contiene cuatro columnas: idcurso: identifica los cursos; denominacion: guarda el nombre de los cursos; fechainicio: registra la fecha de comienzo del curso y fechafin: almacena la fecha cuando se acaba el curso.

Tabla 2: dat_alumno compuesta por cuatro campos al igual que dat_curso: idalumno: identifica a cada alumno del curso; nombrecompleto: guarda el nombre completo de los alumnos; grupo: guarda el grupo académico al que pertenecen; e idcurso: es llave foránea y guarda en que curso está matriculado.

2.1.1.3 Relaciones de muchos a muchos

En este ejemplo un deportista puede practicar varios deportes y al mismo tiempo un deporte puede ser practicado por muchos deportista, generando una nueva tercera tabla debido a la definición expuesta en el epígrafe 1.7.1.2.2 del capítulo 1, dicha nueva tabla nombrada dat_deportistapordeporte.

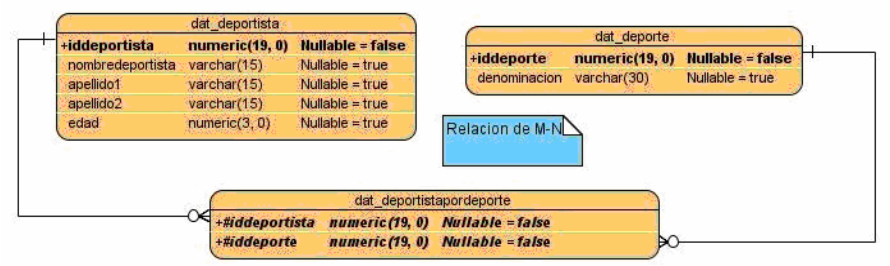

Figura 2.3. Ejemplo de relación de muchos a muchos.

Descripción del diseño:

El diseño está conformado por tres tablas:

Tabla 1: dat_deportista ccntiene cinco columnas: iddeportista: identifica los deportistas; nombredeportista: guarda el nombre de los deportistas; apellido1: registra el primer apellido de los deportistas; apellido2: almacena el segundo apellido de los deportistas, y edad: almacena la edad de los deportistas.

Tabla 2: dat_deporte, compuesta por dos campos: iddeporte: identifica a cada deporte y denominacion: registra el nombre de los deportes que pueden ser practicados por los deportistas.

Tabla 3: dat_deportistapordeporte, esta tabla también llamada tabla de vinculación, contiene la información de los deportistas por deportes, donde su llave primaria está compuesta por las llaves iddeportista e iddeporte que vienen como foráneas de dat_deportista y dat_deporte respectivamente.

Puede darse la situación que la cantidad de deportistas y deportes esté restringida por cuestiones de capacidad u otros intereses del problema. Por ejemplo, en los juegos deportivos existe una cantidad determinada de deportistas por deportes, de manera que por cada uno de ellos debe haber una cifra exacta; por tanto, el diseño en este caso puede ser el mismo (Figura 2.3), sólo que su implementación debe ser validada para que no se inserte en la tabla dat_deportistapordeporte más de lo permitido.

2.1.2 Normalización

2.2.2.1 Primera Forma Normal

A continuación se presenta un ejemplo que no cumple con los parámetros definidos para la primera forma normal, con el objetivo de mostrar la violaciones que puede presentar un diseño y que solución darles.

La primera violación de la primera forma normal se manifiesta en la existencia de atributos multivaluados al tener un atributo llamado color, el cual puede tener varios valores para cada registro. Por otra parte se encuentra el atributo direccion formado por la calle y el número de la vivienda, en este caso el atributo direccion es compuesta por que se quiere obtener un dato específico, como la calle, convirtiéndose así el atributo en compuesto, siendo la segunda violación de la primera forma normal. Debido a las razones antes expuestas el diseño a continuación no se encuentra en primera forma normal. Ver definición de la primera forma normal en epígrafe 1.8.1.3.

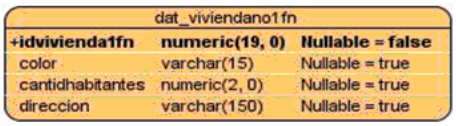

Figura 2.4. Ejemplo de violación de la primera forma normal.

En figura 2.5 se muestra la manera en que debe quedar el diseño correctamente normalizado en primera forma normal creándose tres tablas donde se manipulan correctamente los atributos multivaluados y compuestos, pues como se hace mención, la primera forma normal se define para prohibir grupos repetitivos, que no es más que cuando un atributo puede asumir más de un valor en una misma ocurrencia.

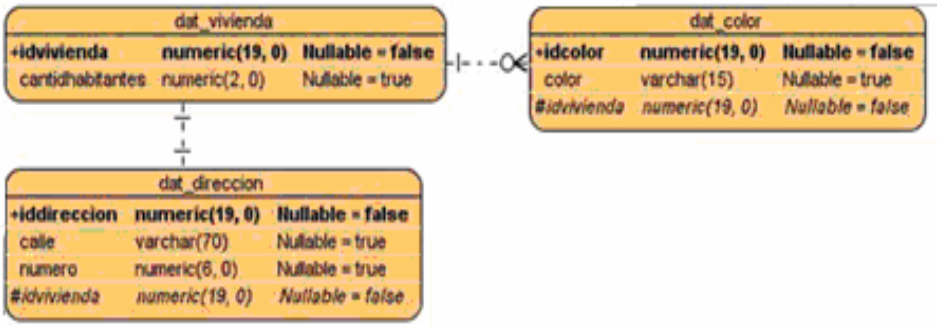

Figura 2.5 Primera forma normal.

Descripción del diseño:

Se ha creado una nueva tabla para la característica color de la vivienda, en donde se guardará todos los colores que puede poseer una determinada vivienda (idvivienda) eliminando así el atributo multivaluado.

Al mismo tiempo se asegura que no existan atributos compuestos con la creación de la tabla dat_direccion, ya que en este caso el atributo direccion está compuesto por la calle y número de la vivienda, ver figura 2.5, por lo que se crea una nueva tabla dat_direccion que contiene estos elementos como columna y se relaciona con la tabla dat_vivienda mediante una relación de 1 a 1, indicando que una vivienda tiene una dirección y así mismo una dirección pertenece a una vivienda.

El diseño expuesto está formado por tres tablas:

Tabla 1: dat_vivienda compuesta por dos columnas: idvivienda: identifica cada vivienda y cantidhabitantes: almacena la cantidad de personas que habitan en la vivienda.

Tabla 2: dat_direccion creada con el objetivo de la utilización de los atributos compuestos, la cual está conformada por con cuatro columnas: iddireccion: identifica cada dirección almacenada; calle: almacena el nombre de la calle de la dirección; numero: guarda el número de la vivienda; e idvivienda: almacena la vivienda a la que pertenece la dirección.

Tabla 3: dat_color creada con el objetivo para la manipulación de los atributos multivaluados, la tabla está formada por tres columnas: idcolor: identifica cada color almacenado; color: guarda el color de la vivienda; e idvivienda: almacena la vivienda a la que pertenece el color.

2.2.2.2 Segunda Forma Normal

Para poder definir la segunda forma normal es necesario saber que es una dependencia funcional, la cual consiste en edificar que atributos dependen de otros atributos.

Recordando la definición de segunda forma normal descrita en el epígrafe 1.7.1.3 del capítulo 1, no es más que cuando una relación cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves), dependen por completo de la clave completa y no sólo de una parte de ella.

Según La Lic. Rosa María Matos plantea que la segunda se hace creando[31]:

- Una relación para todos los atributos que dependen funcional y completamente de la llave (y los atributos que no se analizan por ser atributos llaves, pertenecientes a claves candidatas).
- Otra relación para los atributos que dependan de parte de la llave.

Esto sólo se aplica a relaciones que tienen claves que están formadas por más de un atributo. Si un esquema de relación no está en segunda forma normal, se le puede normalizar a varias relaciones en segunda forma normal, en las que los atributos que dependen de una parte de la clave, formarán una nueva relación que tendrá esa parte de la clave como clave primaria.

En la figura 2.6, se presenta una tabla que está en primera forma normal, pero en la misma no todos sus atributos dependen de la clave primaria en su totalidad, por ejemplo en el caso del atributo cantidad, depende de ambas llaves primarias (idpieza e idalmacen), pero los atributos nombrepieza y paisfabricacion dependen solamente de la clave primaria idpieza, así mismo nombrealmacen y direccionalmacen dependen de la clave primaria idalmacen solamente. Por tanto la tabla dat_piezaalmacen no está en segunda forma normal.

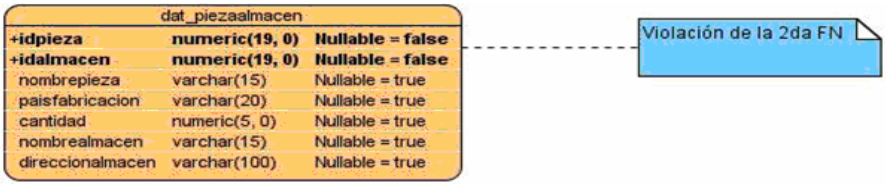

Figura 2.6 Violación de la segunda forma normal.

Representando correctamente la relación de la figura 2.6 siguiendo los pasos de la Lic. Rosa María Matos, se divide la tabla dat_piezaalmacen en dos tablas, en las cuales sus atributos dependen de la clave completa y no sólo de una parte de ella, quedando de la siguiente manera.

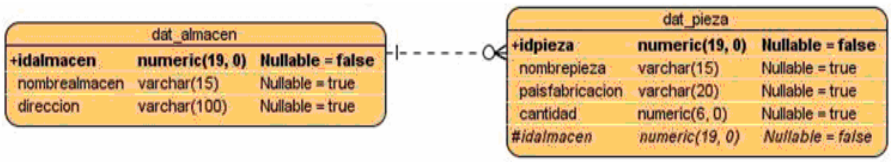

Figura 2.7 Diseño de la segunda forma normal.

Descripción del diseño:

En la figura 2.7 consta de dos tablas dat_almacen y dat_pieza conectadas mediante una relación de uno a muchos, donde un almacén pueda tener muchas piezas y que cada pieza pertenece a un sólo almacén. La relación entre ambas tablas depende del negocio que se esté diseñando, en este caso una pieza puede estar en un sólo almacén, pero bien podría guardarse en varios almacenes, y si fuera la situación la relación sería de muchos a muchos y se agregaría una nueva tabla para tener las piezas por almacenes.

El diseño expuesto está formado por dos tablas:

Tabla 1: dat_almacen compuesta por tres columnas: idalmacen: identifica cada almacén; nombrealmacen: almacena el nombre de los almacenes que se desean registrar en la base de datos; y direccion: almacena dirección de cada almacén.

Tabla 2: dat_pieza contiene cinco columnas: idpieza: identifica cada pieza; nombrepieza: almacena el nombre de las piezas que se desean registrar en la base de datos; paisfabricacion: almacena el nombre del país en que fueron fabricadas las piezas que se desean registrar en la base de datos, e idalmacen: almacena el almacén al que pertenece la pieza.

2.2.2.3 Tercera Forma Normal

La figura 2.8 se encuentra en segunda forma normal, cumpliendo unos de los requisitos que se describen en la definición de tercera forma normal en el epígrafe 1.7.1.3 del capítulo 1, pero se viola el requisito de que en una relación no puede existir dependencia transitiva.

En el caso de los atributos nombreempleado, telefonoempleado, direccionempleado y empresa depende directamente de la llave primaria idempleado, sin embargo el atributo direccionempresa depende de la empresa y a su vez empresa dependa de idempleado dando lugar a que direccionempresa dependa de idempleado pero de manera transitiva.

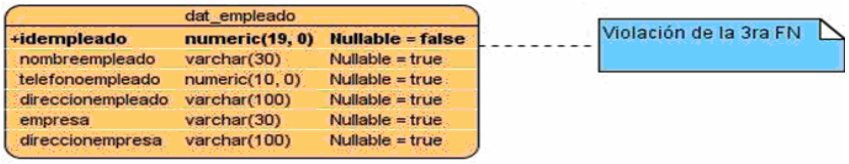

Figura 2.8 Violación de la tercera forma normal.

En la figura 2.9 se muestra el diseño de la figura 2.8 correctamente diseñado cumpliendo con todos los parámetros que se específica para que una relación se encuentre en tercera forma normal.

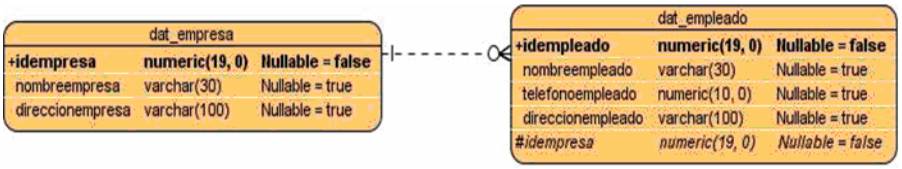

Figura 2.9 Diseño de la tercera forma normal.

Descripción del diseño:

El diseño está conformado por dos tablas:

Tabla 1: dat_empresa está formada por tres columnas: idempresa: identifica cada empresa; nombreempresa: almacena el nombre de las empresas que se desean registrar en la base de datos; direccionempresa: almacena dirección de cada empresa.

Tabla 2 dat_empleado contiene cinco columnas: idempleado: identifica cada empleado; nombreempleado: almacena el nombre de los empleados que se desean registrar en la base de datos;

telefonoempleado: almacena el teléfono del empleado; e idempresa: almacena la empresa a la que pertenece el empleado.

2.2.2.4 Forma Normal Boyce-Codd

La definición de la tercera forma normal puede resultar inadecuada en el caso de una relación donde ocurre lo siguiente[31]:

- La relación tiene varias llaves candidatas.
- Esas llaves candidatas son compuestas.
- Esas llaves candidatas se solapan (o sea, tienen al menos un atributo común).

Para una relación donde no tengan lugar las tres condiciones anteriores, la forma normal de Boyce-Codd es idéntica a la tercera forma normal. Esas tres condiciones son necesarias, pero no suficientes, para que la relación esté en forma normal de Boyce-Codd.

La Lic. Rosa María Matos plantea una relación está en forma normal de Boyce-Codd si, y sólo si, cada determinante es una llave (candidata o primaria)[31]. Obsérvese que se habla en términos de llaves candidatas y no sólo de la llave primaria, ya que una llave es un caso especial de superllave y la llave puede ser candidata o primaria. Además, la definición de forma normal de Boyce-Codd es conceptualmente más simple, aunque es una forma normal más "fuerte". Una relación que esté en forma normal de Boyce-Codd está también en primera forma normal, segunda forma normal y tercera forma normal.

En la figura 2.10 se muestra un ejemplo de un diseño donde la tabla dat_estudianteuci está en la 3FN pero en la misma existen atributos que no son llaves candidatas, por ejemplo: los atributos idestudianteuci, usuario y numeroidentidad son llaves candidatas, sin embargo nombrecompleto y facultad no lo son. Por tanto la tabla no cumple con lo planteado por la Lic. Rosa María Matos y no se encuentra en forma normal de Boyce-Codd.

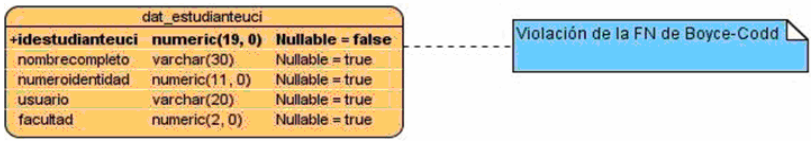

Figura 2.10 Violación de la forma normal de Boyce-Codd.

En la figura 2.11 se muestra el diseño de la figura 2.10 correctamente diseñado cumpliendo con todos los parámetros que se específica para que una relación se encuentre en la forma normal de Boyce-Codd.

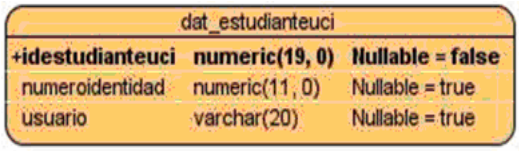

Figura 2.11 Diseño de la forma normal de Boyce-Codd.

En el diseño de la figura 2.11 han excluido aquellos atributos que no son llaves candidatas que se habían incluido para que se entienda cuando no se está en presencia de la forma normal de Boyce-Codd.

Descripción del diseño:

El diseño está compuesto por una tabla:

Tabla 1: dat_estudianteuci contiene tres columnas: idestudianteuci: identifica cada estudiante de la UCI; numeroidentidad: almacena el número de carnet de identidad de los estudiantes que se desean registrar en la base de datos; y usuario: almacena el usuario de cada estudiante.

Se crea un idestudianteuci numérico que es la llave primaria de la tabla, para cumplir con el documento de nomenclatura de la base de datos de Cedrux definido por la Subdirección de tecnología. Este atributo se genera con una secuencia que lo incrementa en 1 cada vez que se inserte un estudiante.

2.1.3 Generalización/ Especialización

Se muestra el diseño de una generalización/especialización, en la cual no se ha se puede aplicar cualquier tipo de restricciones de generalización para controlar la manera en que se va a almacenar la información.

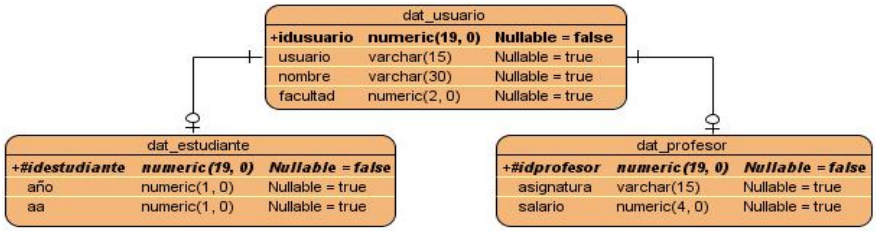

Figura 2.12 Diseño de la generalización/especialización.

El diagrama Entidad – Relación de esta solución, estará compuesto por una entidad general y tantas especializadas se requieran según la cantidad de entidades que compartan atributos en el problema que se esté diseñando. En este caso existe una entidad generalizada y dos especializadas.

Descripción del diseño:

El diseño de la figura 2.12 está compuesto por tres tablas:

Tabla 1: dat_usuario, es la tabla generalizada, la cual está conformada por los atributos comunes de sus hijas. Estos atributos son: idusuario: identifica a cada usuario; usuario: almacena el nombre de usuario en la base de datos; nombre: almacena el nombre completo de los usuarios; apartamentouci: guarda el número del apartamento donde se hospeda en la residencia cada usuario; telefonouci: guarda el número telefónico del apto en que está hospedado y facultad: guarda la facultad a la que pertenece el usuario. Un usuario es, o un estudiantes, o un profesor.

Tabla 2: es la tabla especializada denominada dat_estudiante, está relacionada con dat_usuario a través de una relación de uno a uno, donde un usuario puede ser sólo un estudiante y un estudiante es un usuario. Está compuesta por tres columnas idestudiante: llave foránea proveniente de la tabla usuario e identifica al estudiante; año: almacena el año en que cursa el estudiante en el momento que se inserte en la base de datos y por último aa: guarda si el estudiante es alumno ayudante o no.

Tabla 3: es otra tabla especializada nombrada dat_profesor, la cual está relacionada con dat_usuario, a través de una relación de uno a uno, donde un usuario puede ser sólo un profesor y un profesor es un usuario. La tabla está compuesta por tres columnas idprofesor: llave foránea proveniente de la tabla usuario e identifica al profesor; asignatura: almacena la asignatura que imparte el profesor en el momento que se inserte en la base de datos y por último salario, que guarda el salario que gana el profesor.

2.2 Soluciones para el diseño en Cedrux

2.2.1 Tabla Hash

Las tablas hash en una base de datos relacional, no funcionan exactamente como describe su definición descrita en el epígrafe 1.7.2.1 del capítulo 1, sino que cumplen en gran medida las facilidades que esta brindan. La solución que se muestra en este trabajo, permite guardar los datos que se deseen a partir de una clave asociada a los mismos. Para insertar un elementos se le aplica una función módulo a la clave y se obtiene como resultado el valor hash que se toma como clave primaria de los datos a insertar.

Método para obtener el valor hash: El método utilizado para la obtención del valor hash a partir de la clave fue el Método de División, que consiste en asignarle al valor hash el resto de la división de la clave con un número primo[32].

La función aplicada fue: H(x) = x % 1021. Donde H(x) es el valor hash e identificador de la tabla, x es la clave entrada por el usuario y se ha seleccionado 1021 como divisor, nótese que el hecho de que esté sea un número primo no es algo azaroso, sino que fue cuidadosamente elegido ya que existen

mecanismos que utilizan este tipo de función, con números primos como base, siendo muy comunes en la criptografía. Una vez encontrada una función hash solo queda dar solución al problema de las colisiones.

Resolución de colisiones: Cuando se produzca una colisión se le reasigna otro valor hash a la clave hasta que se encuentre un hueco mediante la función de resolución de colisiones que seria H(x) = x % 1021 +1. A continuación se muestra el diagrama Entidad – Relación que corresponde a la solución:

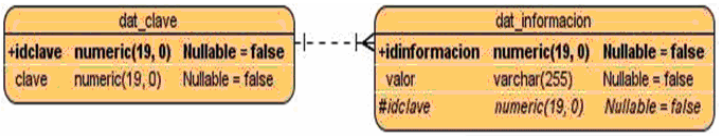

Figura 2.13 Diseño de la solución de tabla hash.

Descripción del diseño:

Tabla 1: dat_clave almacena las claves de la tabla hash, dicha tabla compuesta por dos columnas, idclave: identifica a cada clave; y clave: guarda la clave.

Tabla 2: dat_informacion que guarda los atributos que se quieran guardar a través de la clave que se ha entrada. La tabla está formada por tres atributos el primero de ellos es idinformacion que es llave primaria de la tabla y guarda el valor hash obtenido a partir de la clave, una vez que se le haya aplicado la función hash. Cuenta además con una columna para guardar la clave a la que pertenece el valor. Ambas tablas están enlazadas a través de una relación de uno a muchos pues existe la posibilidad de que para una misma clave se deseen guardar múltiples filas en un determinado momento.

Es importante que se tenga en cuenta que el atributo valor en la tabla hash, puede tener los atributos que se deseen, bastará con agregarlos como columnas a la tabla dat_informacion.

2.2.2 Grafos

A continuación en la figura 2.14 se muestra el diagrama Entidad – Relación de la solución de grafos:

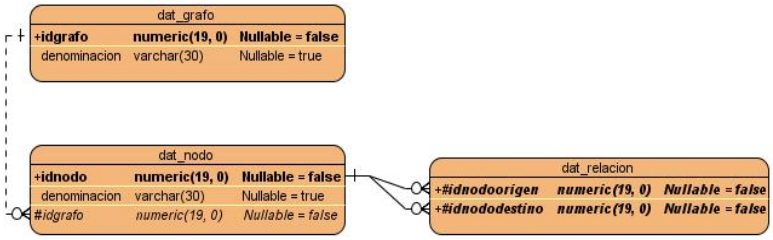

Figura. 2.14 Diseño de solución para Grafos.

Descripción del diseño:

Para la gestión de los grafos en la base de datos, se crean tres tablas relacionadas entre sí de manera tal que describen tres conceptos fundamentales: grafo, nodo y relación.

El diseño está compuesto por tres tablas:

Tabla 1: dat_grafo gestiona los grafos que pueden o no existir en la base de datos. La tabla consta de dos columnas: idgrafo: identifica a cada grafo existente en la tabla y denominacion guarda el nombre del grafo (ejemplo: Grafo1).

Tabla 2: dat_nodo, se encarga de guardar los nodos que componen a cada grafo y tiene tres columnas; idnodo: identifica los nodos del grafo; denominacion: guarda el nombre de estos nodos (ejemplo: Nodo1) e idgrafo: guarda el grafo al que pertenece cada nodo. Las tablas dat_grafo y dat_nodo están relacionadas de uno a muchos indicando que un grafo tiene muchos nodos y que un nodo solo pertenece a un grafo.

Tabla 3: dat_relacion es la encargada de guardar la relación existente entre los nodos de un grafo y de esta manera le da la estructura al grafo y crea los caminos que sean necesarios en este. Esta tabla contiene dos columnas; idnodoorigen: guarda el primer nodo implicado en la relación; idnododestino: guarda el segundo nodo implicado en la relación. Ambos atributos son claves ajenas provenientes de la tabla dat_nodo y forman la llave primaria compuesta de la tabla dat_relacion.

Para la implementación de la solución es necesario:

- Generar las tablas y relaciones a partir del modelo realizado.
- Crear tres secuencias para generar el identificador de cada tabla.
- Crear una función de inserción, actualización y eliminación de los elementos para cada tabla.

2.2.3 Árboles

Se representa los identificadores izquierdos (hijo izquierdo: 2, 4,5) y derechos (hijos: 3,6) y los valores de sus nodos respectivamente y la manera de cómo se organiza el árbol una vez que se han insertado 5 nodos:

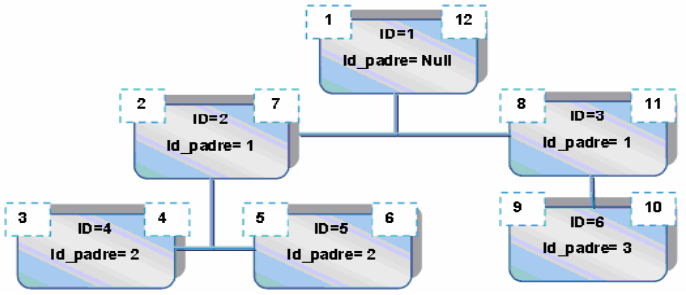

Figura 2.15 Estructura de árbol.

El modelado de una estructura árbol en la base de datos, permite el conocimiento de todos los hijos pertenecientes a un determinado padre y viceversa. A continuación se presenta el diagrama Entidad - Relación.

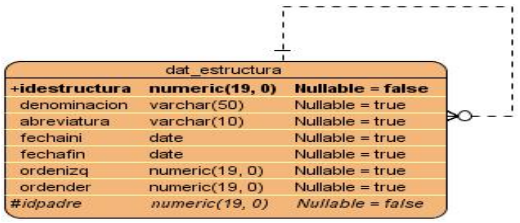

Figura 2.16 Diseño de la solución para árboles

Descripción del diseño:

La tabla dat_estructura cuenta con ocho atributos: idestructura: llave primaria de la tabla e identifica a cada estructura del árbol; denominacion: guarda la denominación de cada estructura del árbol; fechaini: guarda la fecha de inicio de las estructuras; fechafin: guarda la fecha de finalización de cada estructura; ordenizq: guarda el orden de su ubicación en este caso izquierda (hijo izquierdo); ordender: almacena el orden de su ubicación en este caso derecha (hijo derecho); idpadre: guarda el identificador correspondiente al padre de cada estructura.

Los valores de los nodos antes expuestos en la figura 2.15 se determinan mediante una función implementada, que ordena el árbol realizando un recorrido en pre-orden. Comenzará moviéndose el lado externo izquierdo del nodo exterior y luego hacia la derecha.

Este diseño, se puede aplicar a cualquier árbol. Cuando se trabaje con el árbol será de izquierda a derecha, bajando a los hijos de cada nodo, antes de asignar el número de la derecha. Esta forma es llamada algoritmo de atravesamiento de un árbol en pre-orden modificado.

2.2.4 Multientidad

Durante el proceso de investigación se identificó que habitualmente los clientes, artículos, proveedores, formas de pago, y otros, son tablas comunes en las que se puede compartir los datos para evitar el mantenimiento múltiple, pero a la hora de obtener información en el sistema, es necesario poder diferenciarla (o no) desde la perspectiva de una única empresa. Era primordial encontrar una solución a la problemática que se creara a partir de la definición de los propios procesos de negocio del sistema que se pretendía desarrollar. Por ello se diseñó una solución capaz de responder a dicho problema teniendo en cuenta la genericidad del sistema al cual debe beneficiar dicha solución.

Descripción del diseño:

En el diagrama 2.17 se muestran las tres tablas fundamentales que componen la Multientidad:

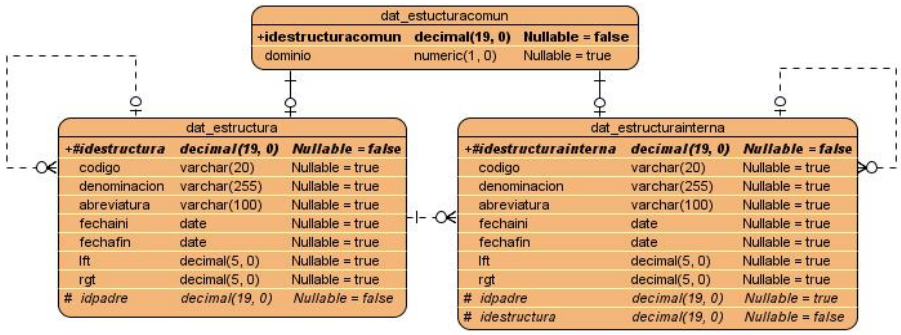

Fig. 2.17 Diseño de la solución para Multientidad

Tabla 1: dat_estructuracomun: guarda la estructura común que será usada en las tablas que sea necesario saber la entidad a la que pertenecen los datos almacenados. La misma posee el atributo idestructuracomun para identificar a la estructura a cual pertenece y el cual será tomado como llave foránea en las tablas que hagan uso de la Multientidad, además tendrá un atributo para guardar el dominio de la estructura.

Tabla 2: dat_estructura: almacena los datos de una estructura general (no interna), la tabla contiene nueve atributos : idestructura: llave foránea de la tabla dat_estructuracomun e identifica a cada estructura en la tabla dat_estructura; el codigo: guarda el código de la estructura; denominacion: almacena el nombre de la estructura; abreviatura: archiva una abreviatura de la estructura; fechaini: almacena la fecha de creación de la estructura; fechafin: archiva la fecha de finalización de la estructura; lft: recoge el orden izquierdo en el árbol de estructuras; rgt: almacena su orden derecho en el árbol de estructuras; e idpadre: almacena el identificador del padre al que pertenece la estructura al tener una organización en forma de árbol.

Estas estructuras pueden tener a la vez cero o muchas estructuras internas expresado en su relación con la tabla dat_estructurainterna.

Tabla 3: dat_estructurainterna tiene los mismos atributos que dat_estructura debido a que las estructuras internas tiene las mismas características de las estructuras generales a las que pertenecen. Un ejemplo: estructura - estructura interna pudiera ser UCI – Facultad 15, donde la UCI sería una estructura y la facultad 15 una de sus estructuras internas.

A continuación en la figura 2.18 se muestra la manera en que se hace uso de la multientidad a través de un diseño:

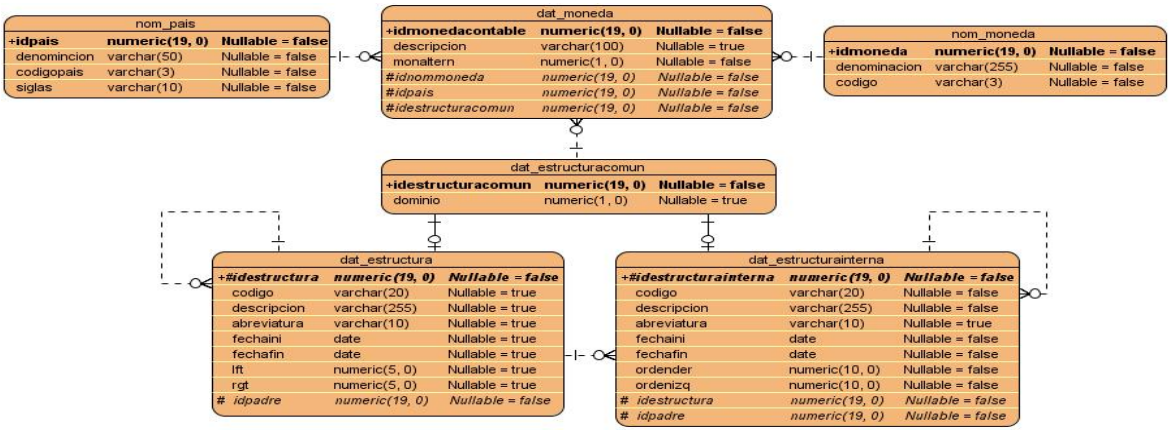

Figura 2.18 Utilización de la multientidad.

Descripción del diseño:

En el diseño figura 2.18 contiene tres de las tablas descritas anteriormente dat_estructuracomun, dat_estructura, dat_estructurainterna en la figura 2.17. Para el uso de la multientidad se han agregado tres tablas más dat_moneda, nom_moneda y nom_pais para mostrar la manera en la que queda un diseño en el que la información en la base de datos sea usada por varias empresas a través de tablas que contienen la información que es común para dichas empresas solamente especificando la empresa o entidad a la que pertenecen en cada uno de los casos.

Este ejemplo contiene la tabla nom_pais que el nomenclador de países para guardar los datos de todos los países, incluye además la tabla nom_moneda que el nomenclador de monedas para guardar los datos de todos las monedas existentes y una tabla denominada dat_moneda para guardar las monedas por países, la estructura a la que pertenecen y otros datos de interés de dichas monedas como por ejemplo una descripción y si es alternativa o no.

La tabla nombrada dat_moneda es donde se usa la multientidad a través del atributo idestructuracomun que es la que guarda la estructura a la que pertenece una moneda determinada de un país específico.

2.2.5 Integración

A continuación se muestra un ejemplo donde es necesario utilizar la integración para resolver el problema planteado en el mismo.

Descripción del ejemplo: es necesario guardar los datos de las reservaciones de la utilería a reservar para la utilización de los directores del Teatro Nacional para la presentación de sus obras, así como del escenógrafo que supervisará la reservación de dichas obras. La utilería que se utiliza en las obras de

teatro se puede definir como todos aquellos objetos que serán utilizados en la puesta en escena de la obra, por ejemplo: trajes, muebles, ambientaciones, pinturas, telas, entre otros. Es usual encontrar en una obra de teatro varios de estos objetos, teniendo en cuenta que la utilería constituye la única con la que cuenta el teatro por lo que será la que las demás obras utilizarán para su puesta en escena. El director de la obra debe reservar con antelación cada utilería que necesitará para su obra. Los escenógrafos son los encargados de supervisar las reservaciones de utilerías realizadas, de manera que una misma persona pueda dar atención a varias reservas, pero cada reservación es atendida exclusivamente por un solo escenógrafo.

Suponga la existencia de tres esquemas organizacionales en la base de datos donde se encontrarán las tablas necesarias para dar solución al problema. Se han identificado 6 tablas nom_director y nom_escenografo pertenecientes al esquema mod_personal, la entidad dat_reservacion en el esquema mod_operaciones, para guardar las reservaciones de utilería realizadas por los directores y supervisadas por los escenógrafos, dat_utiliria y dat_obra en el esquema mod_arte relacionados de manera que una obra puede usar varias utilerías y así mismo una utilería puede ser usada en varias obras, generando una la tabla intermedia o de vinculación dat_utileriaporobra.

Por estar estas tablas en esquemas distintos y estar relacionadas, es necesario considerar una solución de integración de esquemas. A continuación se muestran dos escenarios de integración donde a través de los mismos se puede dar solución al ejemplo planteado.

La figura 2.19 que contiene tres esquemas mod_operaciones, mod_personal y mod_arte, donde para integrar las entidades de estos esquemas basta con relacionarlas de manera directa desde un esquema hacia otro. De esta manera es como se está realizando la integración en la base de datos de Cedrux en estos momentos.

Escenario I

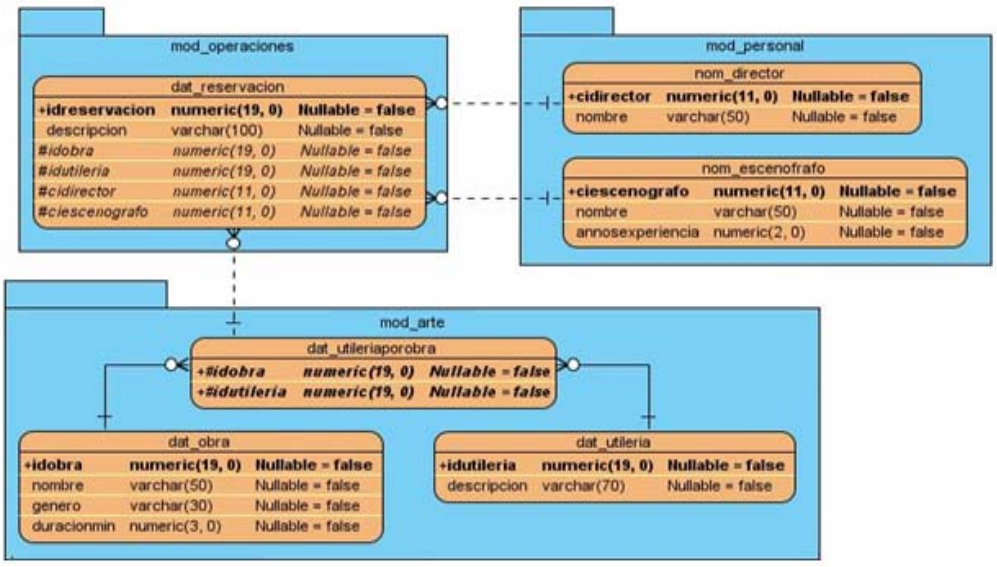

Figura 2.19 Diseño de la solución para Integración (Vieja Cedrux)

A continuación se muestra el Escenario II y en la misma se puede observar que además de los tres esquemas fundamentales se ha sumado un nuevo esquema denominado mod_integracion, donde aparece una entidad nombrada dat_reservacionintegración la cual posee cinco atributos, una llave primaria para identificar cada integración que se realice, que en este caso coincide con cada reservación que sea llevada a cabo por algún director y como se observa es llave foránea proveniente de la relación de uno a uno con la tabla dat_reservacion, el atributo idutileria: guarda la utilería a reservar, idobra: guarda la obra para la que se reserva la utilería, codirector: guarda el director que realiza la reservación y ciescenografo guarda el escenógrafo que supervisará la reservación.

Escenario II

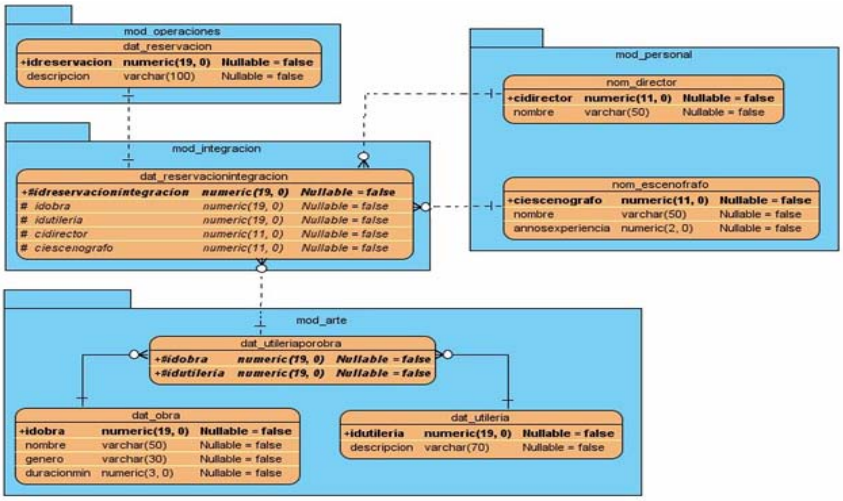

Figura 2.37 Diseño de solución para integración (Nueva)

Este trabajo propone el segundo escenario como solución de integración a emplear en el diseño de la base de datos de Cedrux, pues el primero es candidato a cumplir con las características de flexibilidad, escalabilidad, integridad de los datos y seguridad y protección de los datos, pero no así con la de modularidad, pues al estar relacionados directamente los esquemas mediante las claves ajenas y primarias obliga a existir al esquema que contiene la tabla principal (tabla de la cual proviene la clave ajena).

Por ejemplo en el Escenario I, figura 2.19, la existencia del módulo mod_operaciones depende de la existencia de los módulos mod_arte y mod_personal porque la tabla dat_reservacion contiene atributos que son llaves ajenas provenientes de entidades que se encuentran en estos módulos. Esto prohíbe la instalación de este esquema por sí sólo y niega la característica de modularidad que debe cumplir un sistema ERP.

Por otra parte el Escenario II, figura 2.20, es flexible, escalable, es protegido y seguro, es capaz de asegura la integridad referencial, y es modular pues cada uno de sus módulos puede funcionar por sí sólo. En caso de que se quiera integrar con otro, debe hacerlo a través del esquema de integración que si depende del resto de los implicados en la integración pero no pertenece a ningún módulo de la empresa, es sólo para realizar las integraciones.

2.2.6 Entidad Atributo Valor

La solución del EAV para Cedrux está formada por nueve tablas, a continuación se presenta el diagrama de la solución.

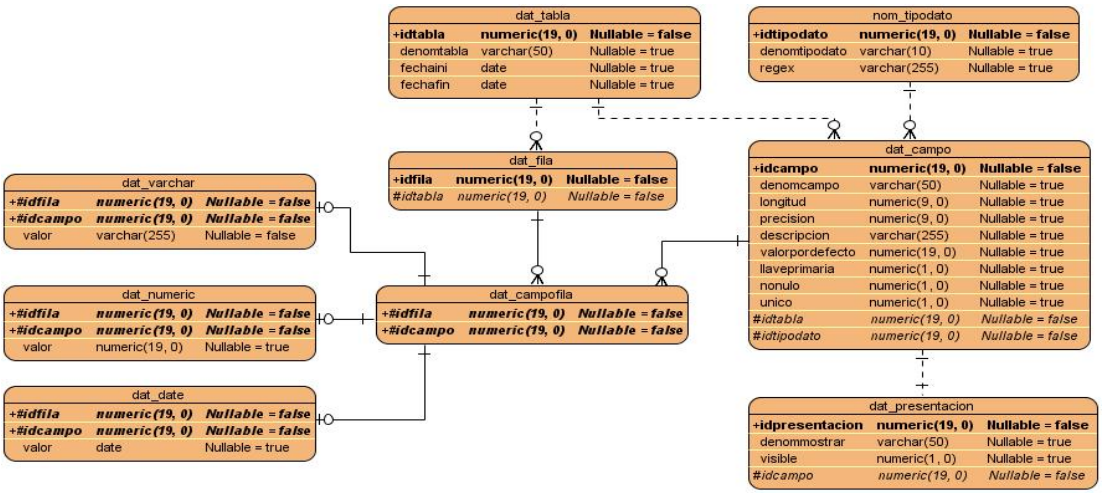

Figura. 2.20 Diseño de la solución para Entidad Atributo Valor

Descripción del diseño:

Tabla 1: dat_tabla se usa para guardar las características de las tablas que se desean crear. Contiene cuatro columnas idtabla: identifica las tablas; denomtabla: guarda el nombre de la tabla; fechaini: guarda la fecha de creación de la tabla, y fechafin: guarda la fecha de eliminación de la tabla.

Tabla 2: dat_fila guarda las filas que pueden componer a una tabla. Está compuesto por dos columnas: idfila: identifica a cada fila; e idtabla: llave foránea que guarda el identificador de la tabla a la que pertenece cada fila.

Tabla 3: nom_tipodato almacena los tipos de datos que existen y que puede tener un campo, en este caso son tres tipos de datos definidos por la Subdirección de Tecnología en el documento de Nomenclatura de base de datos para Cedrux. La tabla presenta tres columnas: idtipodato: identifica los tipos de dato, denomtipodato: guarda el nombre del tipo de dato (Ejemplo: “numeric”), y por último regex para guardar la expresión regular correspondiente a cada tipo de dato.

Tabla 4: dat_campo almacena los campos que pueden componer a una tabla. Se compone por 12 atributos, el atributo idcampo: identifica a cada campo de la tabla; el atributo denomcampo: guarda el nombre de cada campo; longitud: guarda la longitud que puede tener el campo; presicion: guarda los lugares después de la coma que permite el campo si es numérico en caso contrario este valor será NULL; descripción: guarda una descripcion del campo; valorpordefecto: guarda el valor por defecto que recibe el campo; llaveprimaria: guarda si el campo es llave primaria o no; no nulo: guarda si el campo permite nulos; único: guarda si el campo será único o no; idtabla: guarda la tabla a la que pertenece el campo, y por último idtipodato: guarda el tipo de dato que guardará el campo.

Tabla 5: dat_campofila es generada a partir de la relación de muchos a muchos de la tabla dat_fila con dat_campo que es usada para guardar los campos por fila del modelo. La tabla está compuesta por dos columnas: idfila: llave foránea de la tabla dat_fila; idcampo: llave foránea de la tabla dat_campo, ambas conforman la llave primaria de la entidad.

Tabla 6: dat_numeric guarda los valores de los campos por filas que sean de tipo numeric en dependencia del tipo de dato definido para el campo. Esta tabla está compuesta por tres columnas: idfila e idcampo: llaves foráneas de la tabla dat_campofila e identifica a cada valor numérico guardado en la misma, especificando la fila y el campo al que pertenece el dato; valor: guarda la información introducida para un campo numeric determinado.

Tabla 7: dat_varchar guarda los valores de los campos por filas que sean de tipo varchar en dependencia del tipo de dato definido para el campo. Esta tabla está compuesta por tres columnas: idfila e idcampo: llaves foráneas de la tabla dat_campofila e identifica a cada valor numérico guardado

en la misma, especificando la fila y el campo al que pertenece el dato; valor: guarda la información introducida para un campo varchar determinado.

Tabla 8: dat_date guarda los valores de los campos por filas que sean de tipo date en dependencia del tipo de dato definido para el campo. Esta tabla está compuesta por tres columnas: idfila e idcampo: llaves foráneas de la tabla dat_campofila e identifica a cada valor numérico guardado en la misma, especificando la fila y el campo al que pertenece el dato; valor: guarda la información introducida para un campo date determinado.

Tabla 9: dat_presentacion es utilizada para guardar la manera en que se va a mostrar los campos a la lógica de presentación. Está compuesta por cuatro columnas: idpresentacion: identifica la presentación de cada uno de los campos; denommostar: guarda la manera en que se mostrará el nombre del campo; visible: guarda si el campo estará visible a no y por último idcampo guardará el campo que se mostrará.

2.2.7 Control de Concurrencia

Como solución a la concurrencia en la base de datos se creará un atributo denominado versión, el cual será un campo de tipo numérico, siendo el responsable de almacenar la versión en que se encuentra, con el objetivo de que en caso que dos o más usuarios levanten el mismo registro al mismo tiempo para realizar alguna operación de modificación en la base de datos, el primero de ellos que ejecute la acción guardará la versión correspondiente y cuando los demás usuarios ejecuten su acción se les enviará un acuse de recibo indicando que el registro al cual ha accedido ha sido actualizado, que si desea volver a levantarlo para realizar algún cambio en él. Este campo versión se le agregará a todos los recursos o almacenes de datos. Ejemplo:

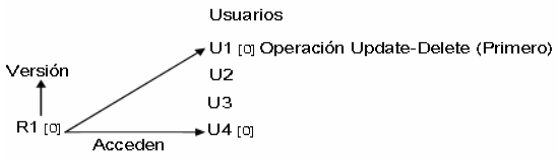

Figura 2.21 Diseño de la solución para concurrencia

Los usuarios U1 y U4 acceden al mismo tiempo al registro, realizando operaciones sobre el mismo, primeramente el registro tiene la versión [0] que le manda al usuario, el primero de los dos usuarios que realice la operación compara la versión que manda es igual a la que tengo si es igual entonces quiere decir que él fue el primero en mandarlo, guarda los cambios y se actualiza la versión.

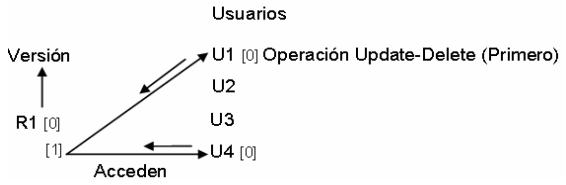

Figura 2.22 Diseño de la solución para concurrencia.

Cuando el usuario U4 quiera mandar la versión que él tiene que en este caso es la [0] todavía, el sistema hará la comparación y como ya la versión cambió mandará un mensaje diciéndole que ya ha sido actualizada o eliminada por otro usuario, y diciéndole si quiere volver a cargar el registro para hacerle su cambio.

Nota: Este es un ejemplo sencillo para solamente dos usuarios en caso de que aumente el número de usuarios sobre un mismo registro se comportará de la misma manera.

A continuación se muestra el diagrama Entidad – Relación de la solución planteada:

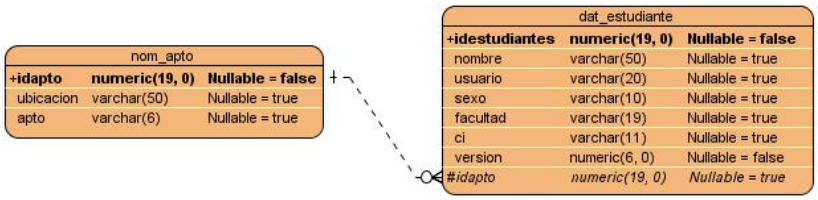

Figura 2.23 Diseño de la solución de concurrencia.

Descripción del diseño:

Como se puede ver en la figura 2.23 se creó un ejemplo que compuesto por dos tablas donde se controla la concurrencia de usuarios en la base de datos con el objetivo de que no se trabaje sobre datos inconsistente. Se crearon una nombrada nom_apto que es un nomenclador que contiene los apartamentos a los que puede pertenecer un estudiante, y otra nombrada dat_estudiante para almacenar los estudiantes que se desean gestionar en la base de datos. En este diagrama se puede ver que en un apartamento viven varios estudiantes, aunque puede que no haya sido asignado a ningún estudiante aún, y un estudiante habita en un único apartamento.

La tabla nom_apto contiene cuatro columnas: idapto: identifica a los apartamentos; ubicacion: guarda la ubicación del apartamento (puede ser el edificio, zona u otra descripción que se quiera guardar); apto: guarda la denominación del apartamento (ejemplo: 125102).

La tabla dat_estudiante está compuesta por ocho columnas; idestudiante: llave primaria de la tabla e identifica a cada estudiante; nombre: guarda el nombre del estudiante; usuario: guarda el usuario del estudiante; sexo: almacena el sexo del estudiante; facultad: guarda la facultad a la que pertenece el

estudiante; version: gestiona la versión de cada tupla de la tabla dat_estudiante para evitar anomalías de acceso concurrente a sus datos ante posibles modificaciones que se estén realizando de manera simultánea por distintos usuarios; ci: guarda en número de carnet de identidad de los estudiantes; e idapto: llave foránea y almacena el apartamento al que pertenece el estudiante.

CONCLUCIONES

En el presente capítulo se ha detallado el diseño realizado, basado en el análisis y la investigación acerca de cada una de las soluciones propuestas, se ha descrito cada entidad presente las mismas mediante ejemplos modelados usando la herramienta CASE Visual Paradigm.

Una vez puntualizado cada uno de los elementos de normalización y buenas prácticas de diseño, así como un grupo de soluciones necesarias para el buen desempeño del diseño de base de datos del Sistema Integral de Gestión Cedrux, estos elementos se encuentran listos para ser sometidos a pruebas de concepto que validen su rendimiento y eficiencia convirtiéndolos en una guía referativa con un gran por ciento de certeza de que la base de datos tiene las características mínimas requeridas para comenzar a dar servicios donde la información que se almacene o se extraiga de la misma sea lo más correcta posible.

CAPÍTULO 3: VALIDACIÓN DE LAS SOLUCIONES. PRUEBAS DE CONCEPTOS

3. Introducción

En este capítulo se realizan las validaciones de las soluciones expuestas en el capítulo dos utilizando pruebas de conceptos.

Se establecieron de 10 a 15 conexiones al servidor de pruebas teniendo en cuenta que este no es un servidor real, pero se simuló que lo fuera para comprobar el rendimiento de las soluciones ante un número determinado de conexiones donde los tiempos de respuesta obtenidos se promediaron obteniéndose la media entre los mismos. A continuación las características del hardware de soporte de las 8 computadoras personal utilizadas para la realización de las pruebas.

Sistema Operativo Windows 7 Ultimate, procesador Intel(R) Core (TM)2 Duo CPU, E4500 @ 2.20GHZ y 1 GB de memoria ram.

3.1 Tabla Hash

3.1.1 Introducción

Esta prueba de concepto se realiza con el objetivo de probar la efectividad y eficacia de la solución de Tabla Hash.

3.1.2 Escenarios

El primer escenario de la prueba de concepto de tabla hash, consiste en una función denominada f_hash_insertar, en la cual recibe como parámetro de entrada una clave entrada por el usuario y la información que se desee guardar bajo dicha clave.

Esta función es la encargada de insertar la información que se desee guardar bajo una determinada clave y en su interior contiene la función hash H(x)= x mod 1021 para generar el índice o identificador del registro a almacenar. El índice se obtiene mediante la selección de la parte entera del cociente, de la división de la clave entrada por el usuario por el número primo seleccionado.

A continuación se presenta la función encargada de la validación de la solución para tabla hash asegurando como plantea su diseño que bajo una clave se pueda guardar varios valores y que cada valor pertenezca a una sola clave.

Función insertar:

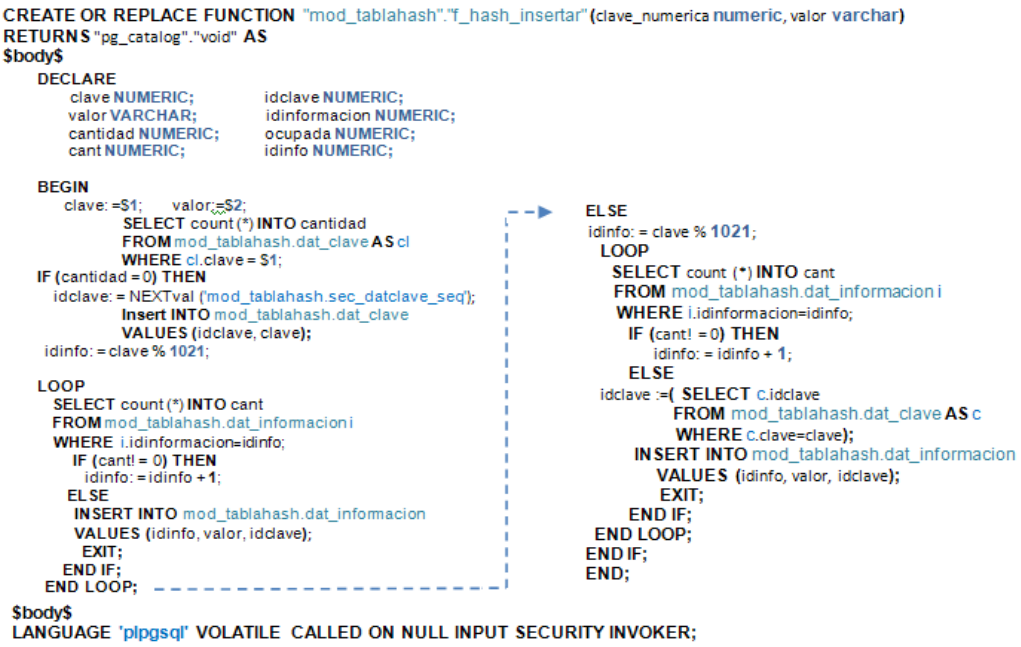

Figura 3.1 Función insertar.

Para validar esta solución se requiere comprobar si a través de una consulta se obtienen los resultados esperados, en tiempos de respuesta razonables.

En la tabla dat_informacion existen cuatro registros guardados bajo la clave 231153 que tiene como identificador 2 y que esta almacenada en la tabla dat_clave mediante la función f_hash_insertar.

idclave	clave
1	110854
2	231153
3	611545
4	876645

Figura 3.2 Tabla dat_clave con datos.

idinformacion	valor	idclave
1	Juan	1
2	Pedro	1
3	Siovel	2
4	Yisel	2
5	Jose Manuel	2
6	Sandro	2
7	Axel	3
8	Makano	3

Figura 3.3 Tabla dat_informacion con datos.

Es necesario comprobar si a través de la consulta de la figura 3.4 los resultados se corresponden con lo mostrado en las figuras 3.2 y 3.3.

Consulta: Obtener la información y el valor almacenado bajo la clave 231153.

```
SELECT i.idinformacion, i.valor
 FROM mod_tablahash.dat_informacion AS i
 INNER JOIN mod_tablahash.dat_clave AS c ON (c.idclave = i.idclave)
 WHERE c.clave=231153
```

Figura 3.4 Consulta.

El resultado de la consulta en la figura 3.4 se muestran en la figura 3.5, comparando visualmente los valores de la tabla 3.5 y la figura 3.2 y 3.3 los valores almacenados bajo la clave 231153 especificada en la consulta de la figura 3.4 son coincidentes.

idinformacion	valor
3	Siovel
4	Yisel
5	Jose Manuel
6	Sandro

Figura 3.5 Resultado de la consulta.

3.1.3 Resultados de la prueba de concepto

A continuación se presenta una comparación del escenario descrito en el epígrafe 3.1.2, primeramente con cuatro registros en la tabla dat_clave y siete registros en la tabla dat_informacion y luego se le agregan 20 000 registro a cada tabla, para observar el comportamiento de la función y de la consulta antes y después de agregarle los 20000, con el objetivo de ver su rendimiento.

	Antes	Después
Función:	0.13 segundos	0.77 segundos
Consulta:	110 milisegundos	140 milisegundos

Figura 3.6 Comparación antes y después de agregarle 20 000 registros.

Se puedo observar que a medida que aumenta la cantidad de registros disminuye el rendimiento de la solución pero en ambos caso el tiempo es relativamente pequeño.

3.2 Grafos

3.2.1 Introducción

Esta prueba de concepto es con el objetivo de probar la efectividad y eficacia de la solución de Grafos.

3.2.2 Escenarios

En el capítulo 2 en el epígrafe 2.2.2 figura 2.14 se muestra el diseño para la solución de grafos. A continuación en la figura 3.7 se presenta el diseño de grafo en la base de datos con sus datos. En el cual un grafo está compuesto por nodos que pueden estar relacionados entre sí.

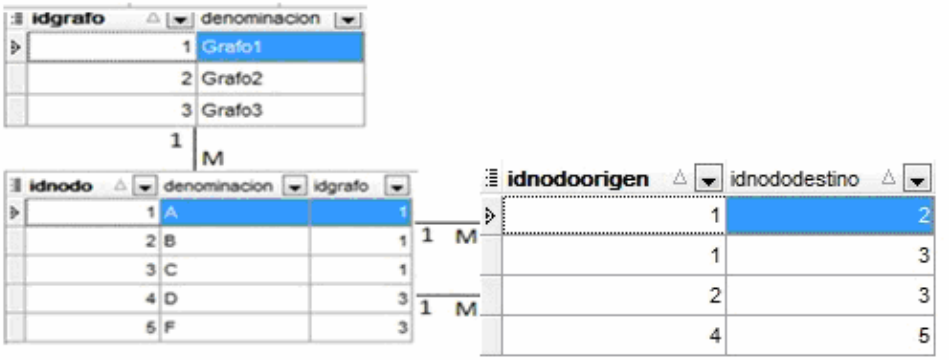

Figura 3.7 Diseño de grafos con datos.

A través de una consulta se puede conocer la relación que guardan estos nodos para un determinado grafo.

Consulta: Obtener el grafo al cual pertenece y la denominación de los nodos que componen las relaciones para un determinado grafo.

```
SELECT go.denominacion AS grafo,
       no.denominacion AS nodo_origen,
       nd.denominacion AS nodo_destino
  FROM mod_grafos.dat_relacion r
       INNER JOIN mod_grafos. dat_nodo no ON (no.idnodo = r.idnodoorigen)
       INNER JOIN mod_grafos. dat_nodo nd ON (nd.idnodo = r.idnododestino)
       INNER JOIN mod_grafos. dat_grafo go ON (go.idnodo = nd.idgrafo)
       INNER JOIN mod_grafos. dat_grafo gd ON (gd.idnodo = no.idgrafo)
WHERE  go.denominacion = 'Grafo1'
```

Figura 3.8 Consulta.

El resultado de la consulta se puede comparar visualmente con la figura 3.7 que muestra el diseño de grafos con datos.

	grafo character var	nodo_origen character var	nodo_destino character var
1	Grafo1	A	B
2	Grafo1	A	C
3	Grafo1	B	C

Figura 3.9 Resultado de consulta de la figura 3.8

3.2.3 Resultados de la prueba de concepto

A continuación se presenta una comparación del escenario descrito en el epígrafe 3.2.2 con los datos inicial del diseño en la figura 3.7 del mismo epígrafe antes y después de agregarle 20000 registros a cada tabla del diseño, con el objetivo de evaluar el rendimiento de la consulta realizada.

	Antes	Después
Consulta:	32 milisegundos	140 milisegundos

Figura 3.10 Comparación antes y después de agregarle 20 000 registros.

3.3 Árbol

3.3.1 Introducción

Esta prueba de concepto es con el objetivo de probar la efectividad y eficacia de la solución de Árboles.

3.3.2 Escenarios

Las funciones f_modificar_padre_estructura, f_eliminar_estructura_hoja y f_reordenar_dat_estructura implementadas para esta solución, son disparadas cada una por un trigger que en su definición específica si es antes o después de alguna modificación que se realice sobre el árbol.

Escenario I: La función f_insertar_estructura es la encargada de insertar las estructuras en la tabla dat_estructura, una vez recibidos como parámetros todos los atributos exceptuando el orden izquierdo y derecho que son generados por otra función que más adelante se describe.

Escenario II: La función f_modificar_padre_estructura tiene como objetivo reasignar un nuevo padre a una estructura determinada. Recibe como parámetros la estructura a la que se le desea cambiar el padre y el nuevo padre que tendrá dicha estructura. Se implementó debido a que en muchas ocasiones se desea eliminar una estructura y si esta es hoja no hay limitantes pero en caso contrario se debe asignar un nuevo padre a todos sus hijos para poder realizar esta operación debido a que PostgreSQL no permite eliminar los registros cuyo identificador esté referenciado en otro como llave primaria, además de que pudiera darse el caso de que esa información se desee conservar o mantener en el mismo árbol. Una vez terminado el proceso de reasignación de padre se procede a eliminar la estructura especificada.

Escenario III: La función f_eliminar_estructura_hoja tiene como objetivo eliminar una estructura hoja en el árbol y tiene como parámetro de entrada el identificador de la estructura que se desea eliminar.

Escenario IV: La función f_reordenar_dat_estructura surge debido a la necesidad de reordenar el orden izquierdo y derecho del árbol de estructuras, ejecutándose después de ocurrir alguna modificación sobre los datos del árbol, permitiendo reordenar los órdenes desde el nivel hoja hasta el nivel cero, utilizando un método de recorrido en pre-orden modificado.

Escenario V: Consulta

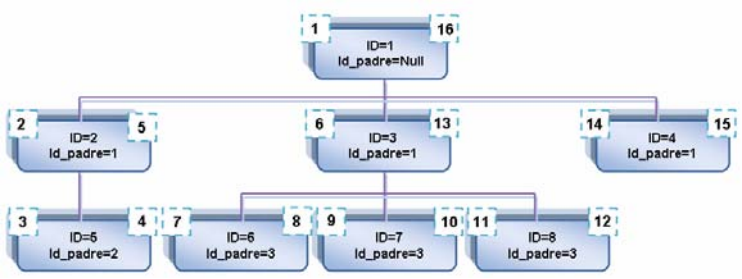

Figura 3.11 Ejemplo de una estructura de árbol

En la figura 3.11 se muestra una tabla con la estructura representada en la figura 3.11 mediante la función f_insertar_estructura. En la misma se puede observar los datos entrados en la función, así como los valores de los atributos ordenizq y ordender, generados por la función f_reordenar_dat_estructura, que es disparada por el trigger t_actualiza_orden_insertar definido para cada fila después de insertar. Ver figura 3.14.

idestructu	denominacion	abreviatura	fechaini	fechafin	ordenizq	ordender	idpadre
1	Estructura1	E1	23/04/2010	20/12/2010	1	16	1
2	Estructura2	E2	23/04/2010	20/12/2010	2	5	1
3	Estructura3	E3	23/04/2010	20/12/2010	6	13	1
4	Estructura4	E4	23/04/2010	20/12/2010	14	15	1
5	Estructura5	E5	23/04/2010	20/12/2010	3	4	2
6	Estructura6	E6	23/04/2010	20/12/2010	7	8	3
7	Estructura7	E7	23/04/2010	20/12/2010	9	10	3
8	Estructura8	E8	23/04/2010	20/12/2010	11	12	3

Figura 3.12 Tabla dat_arbol.

Consulta: Obtener la denominación, la abreviatura, y la fecha de creación de las subestructuras que son hijas inmediatas de la estructura que tiene como abreviatura E3 y que fueron creadas después del 1ro de enero de 2009.

```
SELECT se.denominacion AS estructura, se.abreviatura AS abreviatura,
       se.fechaini AS fecha_fundada
  FROM mod_arboles.dat_estructura AS se
       INNER JOIN mod_arboles.dat_estructura AS e ON (e.idestructura=se.idpadre)
  WHERE e.abreviatura = ' E1' AND se.fechaini > '1/1/ 2009'
  ORDEN BYE se.abreviatura
```

Figura 3.13 Consulta.

El resultado de la consulta 3.13 se puede comparar el resultado con los datos de la figura 3.14 con la figura 3.12

estructura	abreviatura	fecha_fundada
Estructura6	E6	23/04/2010
Estructura7	E7	23/04/2010
Estructura8	E8	23/04/2010

Figura 3.14 Resultado de la consulta 3.13.

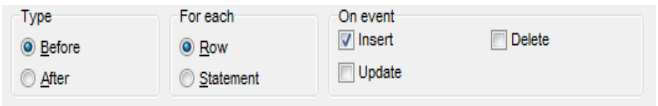

Figura 3.15 Trigger.

3.3.3 Resultados de la prueba de concepto

Escenario I: Función f_insertar_estructura: La función de inserción se ejecutó correctamente y se comprobó que los datos fueran insertados tal y como se introducen en la interfaz de la función. El tiempo de ejecución de la función fue de 0.16 segundos.

Posteriormente se agregaron 20000 registros a la base de datos y se insertó una estructura usando esta función y se observó un aumento en el tiempo de respuesta debido a que para insertar una estructura donde ya existen 20000 obviamente el rendimiento no será el mismo. El tiempo de respuesta promedio de la función de inserción fue de 1.78 segundos

Escenario II: Función f_modificar_padre_estructura: La función para reasignar un padre a una estructura fue comprobada de manera visual y se ejecutó correctamente asignando para la estructura definida en sus parámetros el nuevo padre seleccionado por usuario. El tiempo de ejecución de esta función fue de 0.20 segundos.

El hecho de añadir 20000 registros a la tabla dat_estructura hace que disminuya el rendimiento de la función f_modificar_padre_estructura pero para una función con este tipo de complejidad por el hecho de que primero es necesario encontrar la estructura que tiene como identificador el entrado por usuario y luego se procede a modificar el idpadre de dicha estructura. Este proceso con 20000 registros o más no tendrá la rapidez mostrada en el caso anterior para 8 registros. El nuevo tiempo el mayor que el anterior pero para una solución de este tipo es relativamente rápido, pues pasa levemente los dos segundos con un tiempo de 2.02 segundos.

Escenario III: Función f_eliminar_estructura_hoja: La función que permite eliminar las estructuras hojas se ejecutó satisfactoriamente eliminando la estructura seleccionada por el usuario. Esto se pudo comprobar realizando una consulta a la tabla dat_estructura donde el campo idestructura fuera el mismo que se había eliminado y esta consulta no devolvió ningún registro. El tiempo de ejecución de la función de eliminación de estructuras hojas fue de 0.17 segundos.

Para eliminar una estructura hoja luego de haber añadido 20000 registros a la tabla la función f_eliminar_estructura_hoja demoró un tiempo superior al igual que con las funciones anteriores pues para borrar una estructura es necesario encontrarla a través del identificador especificado y luego se ejecuta la acción de borrado.

Se considera que esta es una solución relativamente rápida al no superar los dos segundos. El tiempo promedio de ejecución de la función es el mostrado a continuación 1.84 segundos.

Escenario IV: Función f_reordenar_dat_estructura: La función para reordenar las estructuras para 8 registros una vez que se realizó una modificación de actualización usando la función f_modificar_padre_estructura asignó un nuevo orden tanto izquierdo como derecho para cada estructura y se comprobó la validez de esta operación de manera visual en la tabla dat_estructura. La ejecución de esta función no alcanzó al segundo con un tiempo de 0.52 segundos.

No ocurrió así para la ejecución de esta función con 20000 registros insertados en la tabla dat_estructura donde luego de modificar una estructura el proceso de actualización desde las hojas hasta la raíz es un tanto más complejo a medida que aumente la cantidad de registros de la tabla. Aún así el tiempo de respuesta en este caso sobrepaso en tan solo 0.05 segundos a los dos segundos y medio, con un tiempo de 2.55 segundo.

Escenario V: Consulta: La consulta realizada devolvió los datos esperados luego de realizar una comprobación de los mismos con la tabla de estructuras insertadas con la función f_insertar_estructura verificando que los resultados de los parámetros de la consulta se corresponden con los existentes en la tabla dat_estructura. La consulta demoró un tiempo de ejecución sobre 8 registros de 47 milisegundos.

Luego de realizada la consulta para 8 registros se agregaron 20000 más a la tabla dat_estructura y se ejecutó nuevamente la consulta para comprobar el tiempo respuesta de la misma con este aumento en el número de registros arrojando un tiempo superior pero relativamente pequeño también, demostrando la eficacia de la solución al devolver los datos esperados en tiempos de respuestas pequeños. El tiempo que demoró la consulta con los 20000 registros agregados fue de 79 milisegundos.

3.4 Multientidad

3.4.1 Introducción

Esta prueba de concepto se realiza con el objetivo de probar la efectividad y eficacia de la solución de Multientidad.

3.4.2 Escenarios

Escenario I: Multientidad

Las figuras 3.16 y 3.17 muestran datos de las tablas dat_estructura y dat_estructurainterna donde se observa las estructuras internas que pertenecen a cada estructura a través de la columna idestructura.

idestructura	fechaini	fechafin	lft	rgt	descripcion	abrevia	codigo	idpadre
1	19/05/2010	10/08/2016	1	2	Empresa de Comunicaciones	ETECSA	100	1
2	30/05/2010	01/01/2020	1	2	Empresa Copextel	COPEXTEL	200	1
3	12/05/2010	20/05/2010	1	2	Misnisterio de Educacion	MINED	300	1

Figura 3.16 Tabla dat_estructura con datos.

idestructuracomun	descripcion	ordender	ordenizq	codigo	idestructura	fechain	fechafin	abreviat	idpadre
3	Sucursal Sancti-Spiritus	1	2	041	1	01/04/2010	Null	S-SSP	3
4	Sucursal Ciego de Avila	1	2	033	1	20/05/2010	Null	S-SSP	4
5	Sucursal Camaguey	1	2	032	1	21/07/2010	Null	S-CMG	5
6	COPEXTEL SSP	1	2	S 041	2	20/05/2010	Null	CPX SSP	6

Figura 3.17 Tabla dat_estructurainterna.

Consulta: Obtener la descripción, la abreviatura, el código, la fecha de creación y el dominio al que pertenecen las estructuras internas de la estructura ETECSA que fueron creadas después del 4 de abril de 2010 y mostrarlas ordenadas por fecha.

```
SELECT ei.descripcion, ei.abreviatura, ei.codigo
       ei.fechaini, ec.dominio
  FROM mod_contabilidad.dat_estructurainterna AS ei
  INNER JOIN mod_estructuracomp.dat_estructura AS e ON (e.idestructura = ei.idestructura)
  INNER JOIN mod_estructuracomp.dat_estructuracomun AS ec ON (ec.idestructuracomun = ei.idestructuracomun)
  WHERE e.abreviatura = ' ETECSA' AND ei.fechaini > '4/04/2010'
  ORDEN BYE ei.fechafin
```

Figura 3.18 Consulta de multientidad.

El resultado de la consulta arrojó los siguientes valores:

descripcion	abreviatura	codigo	fechaini	dominio
Sucursal Ciego de Avila	S-SSP	033	20/05/2010	4
Sucursal Camaguey	S-CMG	032	21/07/2010	5

Figura 3.19 Resultado de la consulta de la figura 3.18.

Como se puede observar en la figura 3.19 la estructura ETECSA pertenecen dos estructuras internas almacenadas en la tabla dat_estructura interna, creadas a partir del 4 de abril de 2010. Los resultados obtenidos se pueden comprobar visualmente a través de la figura 3.17 comparando la correspondencia entre los datos almacenados con los resultados arrojados por la consulta.

Escenario II: Uso de la Multientidad.

Consulta: Obtener la denominación, el código de las monedas, la denominación del país y sus siglas y la estructura a la que pertenecen las monedas ordenadas por dominio de estructura.

```
SELECT nm.denominacion AS moneda, nm.codigo AS codigo_moneda,
       np.nombrepais AS pais, np.siglas AS siglas_pais
       de.descripcion AS estructura, ec.dominio AS dominio_entidad
 FROM mod_datosmaestros.datmoneda AS dm
 INNER JOIN mod_datosmaestros.nom_moneda nm ON (nm.idmoneda = dm.idmoneda)
 INNER JOIN mod_nomencladores.nom_pais np ON (np.idpais = dm.idpais )
 INNER JOIN mod_estructuracomp.dat_estructuracomun ec ON (ec.idestructuracomun = dm.idestructuracomun)
 INNER JOIN mod_estructuracomp.dat_estructura AS de ON ( de.idestructura = ec.idestructuracomun)
 ORDEN BYE ec.dominio
```

Figura 3.20 Consulta.

La consulta de la figura 3.20 tuvo los siguientes resultados:

moneda	codigo_moneda	pais	siglas_pais	estructura	dominio_entidad
Dolar Canadiense	CAD	Canada	CAN	Empresa de Comunicaciones	1
Peso Cubano	CUP	Cuba	CUB	Empresa Copextel	2
Peso Cubano Convertible	CUC	Cuba	CUB	Misnisterio de Educacion	3
Peso Dominicano	DOP	Republica Dominicana	DOM	Misnisterio de Educacion	3

Figura 3.21 Resultado de la consulta de la figura 3.20.

Se puede observar en la figura 3.21 existen varias entidades o empresas a las cuales puede pertenecer una moneda de un país determinado, cumpliendo con la solución para la Multientidad planteada en el capítulo 2, tomando como punto de partida la importancia que tiene en muchas ocasiones la existencia de datos compartidos en una tabla común diferenciados desde la perspectiva de una única empresa.

3.4.3 Resultados de la prueba de concepto

Escenario I: La consulta del Escenario I de validación mostró los resultados esperados luego de haber comprobado que los datos devueltos para las estructuras internas que pertenecen a la estructura ETECSA se corresponden con lo existentes en la figura 3.17 bajo las condiciones definidas en la consulta. El tiempo de ejecución de la consulta fue de 31 milisegundos.

Luego se agregaron 20000 registros a cada tabla, se ejecutó nuevamente la consulta para analizar el comportamiento de los datos devueltos en términos de rendimiento y se comprobó que a medida que aumenta el número de registros en la base de datos el tiempo de respuesta aumenta. El tiempo de ejecución de la consulta con 20000 registros añadidos fue de 93 milisegundos.

Se puede concluir en que el tiempo de la consulta luego de agregados los 20000 registros, aumento con respecto al anterior pero no alcanzó el medio segundo siendo un buen tiempo para una solución con estas características.

Escenario II: La consulta planteada en el Escenario II de validación, arrojó los resultados esperados de manera que se puede diferenciar con facilidad las distintas empresas a las que pertenecen las monedas. El tiempo de ejecución de la consulta fue de 46 milisegundos.

Posteriormente se añadieron 20000 registros a cada tabla y ser ejecutó nuevamente la consulta para ver como se comportaba el rendimiento de la solución a medida que aumenta el tamaño espacial de la base de datos y arrojó como resultado 109 milisegundos.

Como se pudo observar el nuevo tiempo aumentó con respecto al anterior aunque a pesar de contener un mayor número de registros es un tiempo relativamente aceptable para una solución con este tipo de complejidad.

3.5 Integración

3.5.1 Introducción

Esta prueba de concepto es con el objetivo de probar la efectividad y eficacia de la solución de Integración.

3.5.2 Escenario

Para validar la solución se requiere comprobar si a través de una consulta de obtienen los resultados esperados, en tiempos te respuesta razonables teniendo en cuenta que se solicitaran datos que se encuentran en distintos esquemas de la base de datos y se comprobará que estén bien integrados a través de la solución. La figura 3.22 guarda un grupo de datos para cada tabla de los distintos esquemas que se encuentran integrados a través del esquema de integración. Cada una de las tablas implicadas en la consulta contiene 20000 registros donde algunos de ellos se pueden observar a continuación.

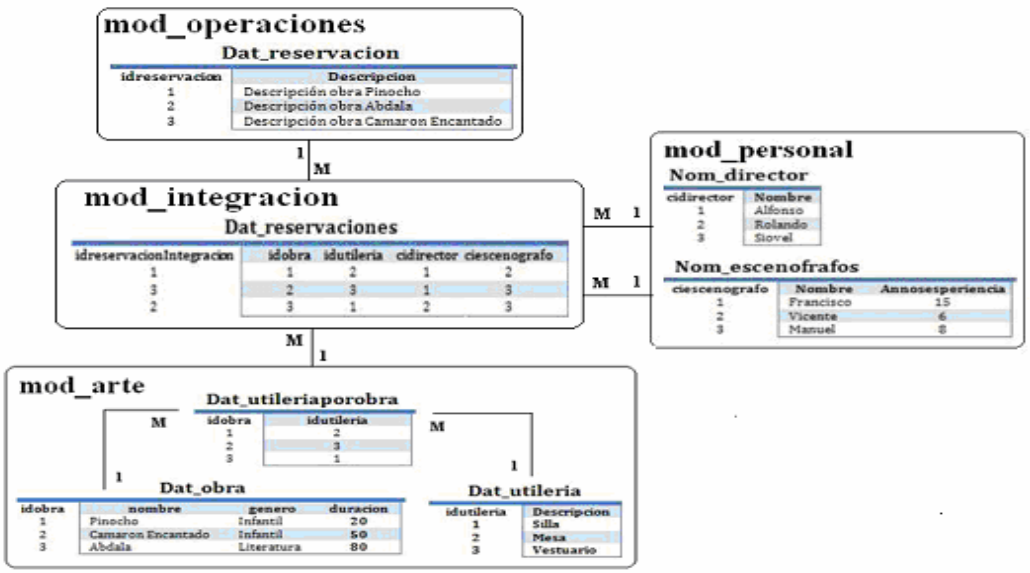

Figura 3.22 Esquemas.

Consulta: Obtener el nombre del director, las obras, el género y la utilería reservadas por un director nombrado Alfonso.

```
SELECT d.nombre AS director, o.nombre AS obra, o.genero,
       u.denominacion AS utilería
  FROM mod_integracion.dat_reservacion AS ri
  INNER JOIN mod_arte.nomutileria AS u ON ( u.idutileria = ri.utileria)
  INNER JOIN mod_personal.nomdirector AS d ON ( d.cidirector = ri.cidirector)
  INNER JOIN mod_arte.dat_obra AS o ON (o.idobra= ri.idobra)
  WHERE d.nombre = 'Alfonso'
```

Figura 3.23 Consulta.

En la consulta 3.23 el nombre del director se obtiene del esquema mod_personal, las obras para las que se reservó utilería al igual que el género están en el esquema mod_arte, es posible conocer la información requerida gracias a la existencia del esquema mod_integracion que contiene las reservaciones y cada uno de los datos necesarios. A continuación en la figura 3.24 se muestran los resultados.

director	obra	genero	utileria
Alfonso	Pinocho	Infantil	Mesa
Alfonso	El Camaron Encantado	Infantil	Vestuario

Figura 3.23 Resultado de la consulta 3.23.

3.5.3 Resultados de la prueba de concepto

Una vez realizada una o varias reservaciones fue posible obtener la utilería utilizada por un determinado director para una determinada obra en presencia de 20000 registros por cada tabla en un tiempo relativamente pequeño, aunque es importante mencionar que a medida que se incremente el número de registros el rendimiento disminuirá, pero en casos excepcionales alcanzará el medio segundo. El tiempo de ejecución de la consulta fue de 110 milisegundos.

3.6 Estructuras Dinámicas EAV

3.6.1 Introducción

Esta prueba de concepto es con el objetivo de probar la efectividad y eficacia de la solución de metadatos.

3.6.2 Escenarios

Escenario I: La función insertar de la solución de metadatos permite insertar tablas, campos mediante las funciones de inserción en las tablas dat_tabla y dat_campo.

La función encargada de insertar un valor en un campo específico de una determinada tabla se denomina f_adicionar_valor y recibe como entradas la tabla a la que pertenece el campo que guardará

el valor, la denominación del campo y el valor, dependiendo del tipo de dato del campo ese valor se guardará en dat_numeric, o en dat_varchar, o en dat_date en dependencia del tipo de datos entrado. Obviamente para insertar un nuevo valor para un determinado campo la función inserta una nueva fila en la tabla dat_fila, indicando la tabla a la que pertenece y una nueva fila por campo en la tabla dat_campofila especificando la fila y el campo.

Escenario II: La función buscar en la solución de metadatos, permite buscar un valor determinado especificando la tabla, el campo y la fila deseada por el usuario. La función encargada de buscar el valor se denomina f_buscar_valor, recibiendo como entrada la tabla donde se desea buscar, el campo del que se quiere buscar un valor y la fila del valor, teniendo como salida el valor que se encuentra en dicha fila para ese campo en la tabla especificada.

Escenario III: Consulta: El conjunto de figuras que presentan a continuación muestra seis de las tablas que componen la solución de metadatos con algunos datos como ejemplo para validar la solución a través de las pruebas de concepto.

idtabla	denomtabla	fechaini	fechafin
1	nom_persona	23/04/2010	23/04/2011
2	dat_reservacion	30/01/2010	24/05/2012
3	dat_trabajador	16/02/2010	24/11/2011

Figura 3.24 Tabla dat_tabla con datos.

idtipodato	denomtipodato	regex
1	numeric	2Ae[[/ep>ux;[S#YQ
2	varchar	3QTKq^a>r)Pc]@!t
3	date	->]u~Ss:^D 7|MK_G

Figura 3.25 Tabla dat_tipodatos con datos.

idcampo	denomcampo	longitud	precision	descripcion	valorpordefecto	llaveprimaria	nonulo	unico	idtabla	idtipodato
1	idpersona	19	0	Llave primaria que identifica a cada persona	0	1	1	Null	1	1
2	nombrepersona	30	Null	Guarda el nombre de la persona	Null	0	0	Null	1	2
3	fechanacimiento	Null	Null	Guarda la fecha de nacimiento de la persona	Null	0	0	Null	1	3

Figura 3.26 Tabla dat_campo con datos.

idfila	idtabla
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1

Figura 3.27 Tabla dat_fila con datos.

idfila	idcampo
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3
4	1
4	2
4	3
5	1
5	2
5	3

Figura 3.28 Tabla dat_campofila con datos.

valor	idfila	idcampo
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1

Figura 3.29 Tabla dat_numeric

valor	idfila	idcampo
Yisel	1	2
Jose Manuel	2	2
Osiris	3	2
Yinet	4	2
Dayannis	5	2

Figura 3.30 Tabla dat_varchar

valor	idfila	idcampo
08/11/1986	1	3
06/11/1986	2	3
20/04/1984	3	3
10/05/1984	4	3
08/08/1990	5	3

Figura 3.31 Tabla dat_date

Consulta: Obtener la denominación de la tabla, la denominación del campo, el tipo de dato del campo, y los valores guardados en el mismo para las filas que están entre la 1 y 4 en el campo nombrepersona perteneciente a la tabla nom_persona.

```
SELECT dt.denomtabla AS tabla, dc.denomcampo AS campo, td.denomtipodato AS tipo_dato
       dv.valor
  FROM mod_metadatos.dat_campo dc
  INNER JOIN mod_metadatos.dat_tabla AS dt ON (dt.idtabla = dc.idtabla)
  INNER JOIN mod_metadatos.dat_campofila AS cf  ON ( cf.idcampo = dc.idcampo)
  INNER JOIN mod_metadatos.dat_fila AS df ON (df.idtabla = dt.idtabla)
  INNER JOIN mod_metadatos.dat_varchar AS dv ON (cf.idfila = dv.idfila)
        AND (cf.idcampo = dv.idcampo)
  INNER JOIN mod_metadatos.nom_tipodatos AS td ON (dc.idtipodato = td.tipodato)
        AND (df.idfila= cf.idfila)
  WHERE dt.denomtabla = 'nom_persona' AND dc.denomcampo = 'nombrepersona'
  AND df.idfila BETWEEN 1 AND 4;
```

Figura 3.32 consulta

Resultado de la consulta:

tabla	campo	tipo_dato	valor
nom_persona	nombrepersona	varchar	Yisel
nom_persona	nombrepersona	varchar	Jose Manuel
nom_persona	nombrepersona	varchar	Osiris
nom_persona	nombrepersona	varchar	Yinet

Figura 3.33 Resultado de la consulta 3.32

3.6.3 Resultados de la prueba de concepto

Escenario I: La función f_adicionar_valor se ejecutó agregando el valor José Manuel en la tabla dat_varchar, teniendo en cuenta que la denominación del tipo de dato del campo nombrepersona en la tabla nom_persona seleccionada por el usuario es varchar. La función se ejecutó en un tiempo de 0.13 segundo.

Luego se realizó la misma operación después de agregarle a cada tabla 20000 registros para comprobar el rendimiento de la solución y aumentó ligeramente el tiempo de ejecución de la función. A pesar del aumento en la cantidad de registros esta función se ejecuta en un tiempo relativamente rápido. El tiempo de ejecución después de agregar los 20000 registros fue de 0.22 segundos.

Escenario II: La función f_buscar_valor devolvió correctamente el valor esperado luego de haber comprobado que ese era el valor guardado para el campo nombrepersona en la tabla nom_persona en la fila 1. La búsqueda se tardó un tiempo de 0.09 segundos retornando el nombre de José Manuel.

Luego se realizó la misma operación después de agregarle a cada tabla 20000 registros para comprobar el rendimiento de la solución y aumentó ligeramente el tiempo de ejecución de la función al igual que en el caso de la función f_adicionar_valor. El tiempo de ejecución de esta función para esa cantidad de registro puede ser considerado eficaz teniendo en cuenta que la diferencia entre este y el

anterior es mucho menor que la diferencia existente entre la cantidad de registros de ambos casos. El tiempo de ejecución después de agregar los 20000 registros fue de 0.23 segundos.

Escenario III: La consulta realizada en el tercer escenario a los datos mostrados (Figura de la 3.24 a la 3.31) devolvió los datos esperados. Se ejecutó la consulta y se comprobó que coincidiera la información arrojada por la misma con los datos que existían en la base de datos. Para devolver la información solicitada la consulta se tardó un tiempo de 62 milisegundos.

Luego se ejecutó la consulta con los 20000 registros agregados en el primer escenario donde el tiempo de respuesta aumentó aunque continuó siendo relativamente pequeño. El tiempo de respuesta en este caso fue de 109 milisegundos.

3.7 Concurrencia

3.7.1 Introducción

Esta prueba de concepto se realiza con el objetivo de probar la efectividad y eficacia de la solución de concurrencia realizada.

3.7.2 Escenario

El escenario consta de dos tablas nom_apto y dat_estudiante relacionadas mediante una relación de uno a muchos donde se verificará el control del acceso concurrente a las mismas.

Para validar la solución se ejecuta una consulta que incluye un inner join, luego se realiza una modificación sobre los datos que devuelve la consulta y una vez modificado algún campo se vuelve a ejecutar la consulta para verificar si el campo versión sufrió alguna modificación.

Consulta: Obtener el nombre, la facultad y la versión de los estudiantes cuya ubicación es el edificio 7.

```
SELECT de.nombre, de.usuario, de.facultad, de.version
  FROM mod_concurrencia.dat_estudiantes AS de
  INNER JOIN mod_concurrencia.nom_apto na ON ( na.idapto= de.idapto)
 WHERE na.ubicacion = 'Edificio  7'
 ORDEN BYE de.idestudiante
```

Figura 3.34 Consulta.

El resultado de la consulta:

nombre	usuario	facultad	version
Jose Manuel	jmbarcelo	15	0
Maikel	msanudo	15	0
Elier	ehernandez	15	0
Leandro	lrojas	15	0
Eduardo	edearmas	15	0
Ricardo	rpvelazques	15	0

Figura 3.35 Resultado de la consulta 3.34.

Función modificar:

```
CREATE OR REPLACE FUNCTION "mod_concurrencia"."f_modificar_facultad_estudiante"
(usuario varchar, facultad_new varchar) RETURNS "pg_catalog"."void" AS
$body$

DECLARE
        usuario VARCHAR;
        facultad_new VARCHAR;
        cantidad NUMERIC;
BEGIN
        usuario:= $1;
        facultad_new:= $2;
SELECT count (*) INTO cantidad
FROM mod_concurrencia.dat_estudiante AS estud
WHERE estud.usuario = usuario;
    IF (cantidad>0)
    THEN
          UPDATE mod_concurrencia.dat_estudiante
          SET facultad = facultad_new
          WHERE mod_concurrencia.dat_estudiante.usuario = usuario;
    ELSE
        Raise EXCEPTION 'Verifique que existe un estudiante con ese usuario';
END IF;
END;

$body$
LANGUAGE 'plpgsql' VOLATILE CALLED ON NULL INPUT SECURITY INVOKER;
```

Figura 3.36 Función modificar.

Después de haber modificado la facultad del estudiantes que tiene como usuario jmbarcelo mediante la función f_modificar_facultad_estudiante, se puede observar en la figura 3.37 que el campo versión ha incrementado su valor en 1 mediante el trigger de la figura que dispara una función denominada chequear.

nombre	usuario	facultad	version
Jose Manuel	jmbarcelo	2	1
Maikel	msanudo	15	0
Elier	ehernandez	15	0
Leandro	lrojas	15	0
Eduardo	edearmas	15	0
Ricardo	rpvelazques	15	0

Figura 3.37 Resultado después de aplicar la función modificar.

En la figura 3.38 se muestra el trigger encargado de que para cada fila entes de actualizar, se dispara la función chequear que contiene en su cuerpo, que es la que se encarga de incrementar en 1 la versión en cada modificación. En este caso se modifica una sola fila pero en caso de que fuera más de una se aumenta cada versión de las filas involucradas y modifica el o los campo deseados.

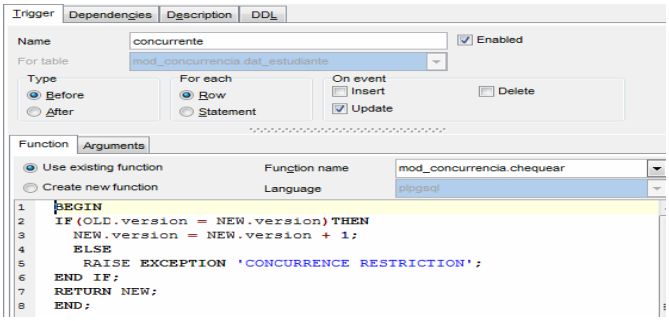

Figura 3.37 Trigger.

3.7.3 Resultados de la prueba de concepto

La consulta realizada sobre las tablas nom_apto y dat_estudiante con 20 y 6 registros respectivamente devolvió los datos esperados después de haberlos comparados con los datos iniciales de la figura 3.35, demorado un tiempo de 32 milisegundos.

El tiempo de la función modificar f_modificar_facultad_estudiante tardó un tiempo de 0.19 segundos. Posteriormente se agregaron 20000 registros a cada una de las tablas implicadas, donde se ejecutó la consulta y la función para observar el rendimiento de ambas arrojando un tiempo de 78 milisegundos y 0.23 segundos respectivamente.

Como se puede observar en los resultados devuelto por el Sistema Gestor de Base de Datos, los tiempos después de haber agregado 20000 registros a la solución son mayores pero aun así son relativamente pequeños.

CONCLUSIONES

En este capítulo se han analizado varios elementos que demuestran la validez de cada una de las soluciones del diseño de la base de datos de Cedrux, teniendo en cuenta los tiempos de respuesta de consultas para extraer información de la base de datos, tiempos de ejecución de varias funciones que son necesarias para el uso de algunas de estas soluciones, comprobación visual de la información arrojada por cualquier acción sobre los escenarios de prueba en cada una de las soluciones, tomando como punto de partida imágenes de los datos almacenados en las tablas y comprobando su coincidencia con los que muestran las respuestas del sistema en cada escenario.

Las pruebas realizadas en este capítulo hacen de las soluciones diseñadas en este trabajo una guía referativa para el diseño de base de datos del Sistema Integral de Gestión Cedrux y facilita el trabajo de los arquitectos de datos ganando en tiempo y logrando eficacia en sus diseños para los distintos esquemas de la base de datos.

CONCLUSIONES GENERALES

Una vez terminada la investigación se considera que el presente trabajo ha cumplido con los objetivos propuestos.

- Se realizó una investigación de los elementos teóricos del diseño de una base de datos, información que posibilitó el desarrollo de la investigación.
- Se obtuvieron modelos de datos correspondientes a elementos que constituyen buenas prácticas para lograr un diseño eficaz, y a un conjunto de soluciones comunes para los distintos esquemas, necesarias para cumplir con los requerimientos del sistema y evitar la incoherencia de los diseños en los diferentes módulos de la base de datos de Cedrux
- .A partir de los modelos de datos se aplicaron escenarios de prueba para validar la solución mediante las pruebas de concepto permitiendo evaluar la eficacia de las mismas.

El resultado de esta investigación permitirá una organización y un aumento de la calidad en la base de datos del proyecto Cedrux, a partir del empleo de las soluciones y las buenas prácticas propuestas, donde los arquitectos de datos del proyecto tendrán un documento rector para que así se gane tiempo y exista una coherencia entre los distintos módulos que lo componen.

RECOMENDACIONES

Se recomienda que se añada a la guía el tema referente a la nomenclatura para el diseño de base de datos del Sistema de Gestión Cedrux, que hoy se encuentra especificada en un documento generado por la Subdirección de Tecnología, con el objetivo de que cada diseño que se realice, cumpla con los estándares de nomenclatura definidos por la subdirección.

Se recomienda incluir a este trabajo una solución para la gestión de la internacionalización o multilenguaje del Sistema de Gestión Cedrux con el fin de garantizar la personalización del sistema de acuerdo al idioma en el que se desee visualizar las vistas de la lógica de presentación.

Se recomienda agregar a la guía una solución para almacenar patrones de diseño que sean usados en el desarrollo del Sistema de Gestión Cedrux, con el fin de que sirva de ejemplo a los arquitectos de datos del proyecto ERP para la realización de esta tarea.

REFERENCIAS BIBLIOGRÁFICAS

1. MARQUÉS ANDRÉS, M. Apuntes de Ficheros y Bases de Datos. Universitat Jaume I. 2001. [Consultado el: 3 de mayo de 2010]. Disponible en: http://www3.uji.es/~mmarques/f47/apun/apun.pdf.

2. GRAYSON, T. Diseño de bases de datos relacionales: principios básicos de diseño. MitOpenCourseWare. Massachusetts institute of technologic universia.Disponible en: http://mit.ocw.universia.net/curso11208/11/11.208/IAP02/lecture-notes/lecture5-2.html#Qualities_of_a_Good_Database_Design.

3. Bases de Datos. Information & Technologic Management, ITM Disponible en: http://www.i-t-m.com/productos-servicios/bases-de-datos.

4. RODAS, C. y SAMANIEGO, G. Base de Datos. Diseño, Normalización y Elaboración de una Base de Datos. [Consultado el: 15 de mayo de 2010]. Disponible en: http://www.slideshare.net/ChristianR/diseo-de-base-de-datos.

5. FALLAS, J. Sistemas integrados de información Geográfica. Diseño de base de datos. Laboratorio de Teledetección y Sistemas de Información geográfica. Universidad Nacional. Heredia. Costa Rica. 1997.Disponible en: http://www.mapealo.com/Costaricageodigital/Documentos/alfabetizacion/Diseno_bases_datos.pdf.

6. COSTAL COSTA, D. Introducción al diseño de base de datos Universitat Oberta de Catalunya. Clases de pregrado. 2007.[Consultado el: 4 de abril de 2010]. Disponible en: http://ocw.uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02150.pdf.

7. RAMOS, Y. A.; NAPOLEÓN, C.*, et al.* Análisis y desarrollo de un sistema de activo fijo para la adecuada administración de los activos institucionales del ministerio de gobernación. Universidad Andrés Bello, Facultad de Ciencias Económicas. Tesis para optar por el título de Licenciado en Ciencias de la Computación. 2008.[Consultado el: 15 de mayo de 2010]. Disponible en: http://www.unab.edu.sv/bvirtual/13900/documentocompleto.pdf.

8. CHAMBAS ERAS, L. A. Diseño y Gestión de base de datos. UNIVERSIDAD NACIONAL DE LOJA, Ingeniería en Sistemas. Clases de Pregrado. 2008.[Consultado el: 2 de junio de 2010]. Disponible en: http://eqaula.org/eva/file.php/blog/attachments/409/Capitulo2.pdf.

9. ROB, P. y CORONEL, C. *Sistemas de bases de datos: diseño, implementación y administración.* 5 ed. 838 p. ISBN 9706862862.

10. KAEDU. Gestión y Modelamiento de Base de Datos. [Consultado el: 3 de junio de 2010]. Disponible en: http://xuaxpedia.blogspot.com/2009/10/base-de-datos-conceptos-basicos-parte2.html.

11. MONGE, H. Bases de Datos Facultad Multidisciplinaria Paracentral, Clases de Pregrado. 2010.([Consultado el: 3 de marzo de 2010]. Disponible en: http://monbarher.orgfree.com/bad175.htm.

12. CHÁVEZ, C. A. G. Diseño de bases de datos relacionales. nº Disponible en: http://www.mailxmail.com/curso-diseno-base-datos-relacionales.

13. QUIROZ, J. El modelo relacional de bases de datos. *Boletín de Política Informática*, 2003, nº 6, Disponible en: http://www.inegi.org.mx/inegi/contenidos/espanol/prensa/Contenidos/Articulos/tecnologia/relacional.pdf.

14. FLORES, J.; LARIN, A.*, et al.* IMPLANTACIÓN DE UNA APLICACIÓN WEB PARA OPTIMIZAR EL ENLACE LABORAL DE LA CÁMARA DE COMERCIO E INDUSTRIA DE EL SALVADOR, FILIAL SAN MIGUEL. Tesis para optar por el título de Ingeniero en Sistemas Informáticos. UNIVERSIDAD DE ORIENTE, FACULTAD DE INGENIERÍA Y ARQUITECTURA, SAN MIGUEL, EL SALVADOR. 2009.Disponible en: http://www.univo.edu.sv:8081/tesis/020112/.

15. ¿Qué son los ERP? Instituto Ibermática de innovación. [Consultado el: 15 de mayo de 2010]. Disponible en: http://www.i3b.ibermatica.com/i3b/lqsqs/lqsqs_queeserp.pdf/download.

16. TOLEDO, L.; PERURENA, L.*, et al.* Herramienta para la enseñanza del Modelo Conceptual de Bases de Datos utilizando las Nuevas Tecnologías de la Información y las comunicaciones. UNIVERSIDAD DE CIENFUEGOS "CARLOS RAFAEL RODRÍGUEZ", FACULTAD DE INFORMÁTICA.[Consultado el: 10 de mayo de 2010]. Disponible en: http://www.bibliociencias.cu/gsdl/collect/revistas/index/assoc/HASHa2c7/1fa93c61.dir/doc.pdf.

17. STORTI, G.; RÍOS, G.*, et al.* Base de Datos. Modelo Entidad Relación Colegio Manuel Belgrano. Clases de Pregrado de la asignatura "Tecnología de la Información y la Comunicación". . [Consultado el: 3 de junio de 2010]. Disponible en: http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf.

18. CASELLES, J. Proceso de Normalización Curso del lenguaje SQL. 2008.Guadalajara SQL Iniciación, Lenguaje SQL y SQL Server.Disponible en: http://www.conínteres.es/sql/material/Proceso_de_Normalizacion.pdf.

19. Normalización. Clases de Pregrado.Base de Datos.Universidad de las Ciencias Informáticas.Disponible en: http://eva.uci.cu.

20. C.J.DATE. *Introducción a los Sistemas de base de datos.* Editado por: Pearson Educación. 2001. vol. 7Mo, ISBN 0-201-38590-2.

21. GÓZALES, V. G. Tutorial de base de datos. Características del modelo E-R extendido Disponible en: http://www.victorgarcia.org/pfc/modeloER/extendedER.php.

22. GIRALDO, L. y ZAPATA, Y. HERRAMIENTAS DE DESARROLLO DE INGENIERIA DE SW PARA LINUX Disponible en: http://hugolopez.phi.com.co/docs/download/file=Giraldo-Zapata-Herramientas%20de%20ISW.pdf,_id=17.

23. PostgreSQL Tools.Products Family. SQL Manager.net Disponible en: http://www.sqlmanager.net/products/postgresql/.

24. FABIÁN, J.; YSAAC, J.*, et al.* PARTICIONAMIENTO HASH. Base de Datos II. Informe. Trabajos/Información UNSA.J. Ysaacx M.A. IX Semestre. Disponible en: http://psicosix.iespana.es/bd2/trabajo01.pdf.

25. Base de datos orientada a grafos Wikipedia. Disponible en: http://es.wikipedia.org/wiki/Base_de_datos_orientada_a_grafos.

26. TDA no lineales. Curso de Pregrado. Programación 2.Universidad de las Ciencias Informática. Disponible en: http://eva.uci.cu.

27. Definición de Disponible en: http://definicion.de/integracion/.

28. PLANEACION DE RECURSOS EMPRESARIALES (E.R.P.) Trabajos de Contabilidad. Disponible en: http://trabajos-contabilidad.blogspot.com/2009/04/planeacion-de-recursos-empresariales.html.

29. BALTEUS. Alternativas EAV con XML (PostgreSQL 3) Disponible en: http://balteus.blogspot.com/2009/05/alternativas-eav-con-xml-en-postgresql.html.

30. AZCÁRATE, E. Q. PostgreSQL Como funciona una Base de Datos por dentro. Disponible en: http://wiki.postgresql.org/images/4/43/Postgresql_como_funciona_una_dbms_por_dentro.pdf.

31. GARCÍA, R. M. M. *Diseño de bases de Datos.* ISBN 959-13-1273-3.

32. QUEVEDO, E.; ASENCIO, A.*, et al.* Programación. Tema 8: Tablas Hash Escuela Técnica Superior de Ingenieros de Telecomunicación Departamento de Ingeniería Telemática. Universidad de Las Palmas de Gran Canaria.2005.Disponible en: http://www.iuma.ulpgc.es/users/jmiranda/docencia/programacion/Tema8_ne.pdf.

GLOSARIO

C

Campo o atributo: Es la unidad menor de información sobre un objeto (almacenada en la base) y representa una propiedad de un objeto (por ejemplo, el color).

D

Dominio: Conjunto de valores de los cuales los atributos obtienen sus valores.

E

Esquema: Describe la estructura de una Base de datos, en un lenguaje formal soportado por un Sistema administrador de Base de datos (DBMS). En una Base de datos Relacional, el Esquema define sus tablas, sus campos en cada tabla y las relaciones entre cada campo y cada tabla.

Entidad: Representación de un objeto individual concreto del mundo real. Cualquier tipo de objeto o concepto sobre el que se recoge información puede ser una cosa, persona, concepto abstracto o suceso.

G

Guía: Ostenta diversos significados de acuerdo al contexto en el cual se lo aplique. En términos generales, es aquello o a aquel que tiene por objetivo y fin el conducir, encaminar y dirigir algo para que se llegue a buen puerto en la cuestión de la que se trate. La Real Academia de la Lengua Española la define como aquello que dirige o encamina.

Guía Referativa: Es aquello que se puede tomar como referencia con el fin dirigir o encaminar la realización de un conjunto de acciones, algún proceso, evento, o persona.

L

Llave Primaria: Llave con valores únicos, es decir, no ocurren más de una vez en el atributo.

Llave Foránea (foreign key o clave foránea): Es aquella columna que existiendo como dependiente en una tabla, es a su vez clave primaria en otra tabla.

R

Registro: Hilera o fila en una tabla. También se conoce como tupla.

Referativa: Es utilizada con la intención de indicar que aquello que lleve su nombre se puede emplear como referencia para la realización de algún fin determinado.

S

Secuencia: Es un generador secuencial de números con nombre. Son utilizadas frecuentemente para crear claves artificiales. Pueden ser configuradas para incrementarse, decrementarse y repetirse.

Servidor: Un servidor, en informática o computación, es el ordenador en el que se ejecuta un programa que realiza alguna tarea en beneficio de otras aplicaciones llamadas clientes.