**Universidad Central "Marta Abreu" de Las Villas.**
**Facultad Matemática, Física y Computación**
**Ingeniería en Informática**


# TRABAJO DE DIPLOMA

# Herramienta CASE DWH-Inspector para el Análisis de Almacenes de Datos.

***Autores:*** **Ashley Gálvez Díaz**
**Pedro Luís Torres Portal**

***Tutores:*** **Dr. Rosendo Moreno Rodríguez**
**Lic. Lindsay Gómez Beltrán**

***Santa Clara, 2010***
***"Año 52 de la Revolución"***

## Dictamen

Hago constar que el presente trabajo fue realizado en la Universidad Central “Marta Abreu” de Las Villas como parte de la culminación de los estudios de la especialidad de Ingeniería Informática autorizando a que el mismo sea utilizado por la institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos ni publicado sin la autorización de la Universidad.

_____

Firma de los autores

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdos de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del tutor

Firma del Jefe del Seminario de Programación

## *Pensamiento*

*Empieza por hacer lo necesario, luego lo que es posible y de pronto te encontrarás haciendo lo imposible.*

*San Francisco de Asís*

# *Agradecimientos*


*Ashley Gálvez Díaz.*

*Muchas han sido las dificultades que he enfrentado, en este largo camino como estudiante, de este modo quisiera agradecer a todas aquellas personas que han aportado su ayuda para alcanzar este logro.*
*Muchas gracias a:*
*Todos los que de una forma u otra han contribuido en mi formación.*
*Aquellos que han dado parte de su vida a mi preparación como persona y en especial a mi papá.*
*A los que han hecho posible la realización de este trabajo.*

*Pedro Luís Torres Portal.*

*A mis abuelos Dora y Rafael que el destino no les permitió estar aquí pero que fueron dos personas muy especiales en mi vida.*

*A mi mamá que lo es todo para mí, que la quiero mucho y que sin ella no fuera nada.*

*A mi papá por haberme apoyado siempre.*

*A mi padrastro por haber estado a mi lado desde que era pequeño.*

*A mis tíos Lourdes y Rafelito por haberme ayudado y apoyado tanto en mi vida.*

*A mi novia que no pudo estar aquí, pero que es alguien muy importante para mí.*

*A todos mis amigos y personas que de una forma u otra han contribuido a que yo estuviera donde estoy.*

# *Dedicatoria*

*Dedicamos este trabajo a todos los que nos han apoyado.*

*A nuestros abuelos, padres y familiares.*

*A la Revolución, al Comandante Fidel Castro y a la UCLV.*

## Resumen

El trabajo de Análisis y Modelado para Almacenes de Datos, específicamente aplicando el Modelo Estrella o el de Copo de Nieve, en varias ocasiones se torna complicado, puesto que no se poseen herramientas comerciales que generen estos modelos adecuadamente para el almacén a partir de sistemas de información legados, que generalmente existen en base al modelo relacional. Teniendo en cuenta esta dificultad se plantea en este trabajo la creación de una primera versión de una herramienta CASE de Análisis y Diseño de sistemas de información para la toma de decisiones, que facilite el trabajo de analistas, implementadores y usuarios, al revisar automatizado los metadatos estructurales de las Bases de Datos relacionales existentes, y proponer una estructura dimensional óptima para el Almacén. Este software debe ser simple de usar hasta para los usuarios que no tengan todo el conocimiento y experiencia requerido sobre el tema.



### PALABRAS CLAVE

Almacenes de Datos, Modelado Dimensional.

## Abstract

The Analysis and Modeling for Data Warehouses, specifically applying the Star Model or Snowflake Model, in several occasions is complicated, since are not possessed commercial tools that generate these models appropriately for the warehouse from legacy information systems that generally exist based on the relational model. Keeping in mind this difficulty, in this work thinks about the creation of a first version of a CASE-tool for Analysis and Design of information systems for the decisions support, that facilitates the work of analysts, programmers and users, when revising automated the meta-structural of the existent Relational Databases, and to propose a optimal dimensional structure for the Warehouse. This software should be simple for using, until for the users that don't have the whole knowledge and experience required on the topic.



### KEY WORKS

Data Warehouse, Dimensional Model.

# ÍNDICE

# Introducción

# Introducción

Con el gran desarrollo de la Informatización, crecen las ventajas para quienes tienen acceso a grandes volúmenes de datos, pero a su vez esto trae un gran problema ¿cómo manejar de manera exitosa estos volúmenes?

Hoy en día existen diversos tipos de sistemas de soporte para la toma de decisiones, pero el que ha tenido más auge a escala mundial en las grandes instituciones sin duda ha sido el *Data Warehouse* o Almacenes de Datos, convirtiéndose en el centro de atención de las organizaciones, puesto que provee un ambiente para hacer un mejor uso de la información administrada por diversas aplicaciones operacionales.

El almacén de datos no es más que una colección de datos orientada a un determinado ámbito (empresa, organización, etc.), integrado, no volátil y variable en el tiempo, que ayuda a la toma de decisiones en la entidad en la que se utiliza. Se trata, sobre todo, de un expediente completo de una organización, más allá de la información transaccional y operacional, almacenada en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos (Inmon 2005).

En el caso de las empresas cubanas la introducción de tecnologías de la información y las comunicaciones en el manejo del conocimiento tiene como objetivo fundamental el uso más racional de los recursos, el logro de una mayor productividad y la obtención de los productos con una mayor calidad. Aunque no se cuenta con gran experiencia en el uso de los *Data Warehouse* si existen empresas que han tenido experiencias en su uso principalmente en el turismo.

En Cuba se ha llevado a cabo un intenso proceso inversionista y de formación de personal, que permite afirmar que existen varios especialistas, no solo graduados de carreras afines a la informática, en diferentes organismos e instituciones dedicadas al desarrollo de sistemas de información, que trabajan o han trabajado en la implementación y explotación de almacenes o mercados de datos a partir de varios sistemas históricos de información y que después aplican diferentes herramientas estadísticas y de búsqueda de información no revelada tradicionalmente en esos sistemas, para detectar tendencias industriales y comerciales, etc.

En la propia UCLV se han desarrollado varios trabajos aplicados a diferentes esferas, en calidad de tesis de maestrías y de diploma que abordan la creación de almacenes de datos. En todos se nota que el estudio de los metadatos o estructura de los sistemas fuentes se realiza de forma manual, y el analista o desarrollador propone un modelo dimensional (esquema estrella o esquema copo de nieve) adecuado para los requerimientos del usuario. A partir de ahí la mayor atención en el desarrollo siempre ha estado en la creación de las herramientas o software de limpieza, transformación y carga de los datos desde los diferentes sistemas de información al almacén; o en lo correspondiente a la explotación de este con la creación de informes

generalmente de resúmenes variados, o proponiendo herramientas de creación de estas consultas o procedimientos que permiten generar dichas tablas o gráficos resultantes.

Es entonces que se considera necesario dar alguna solución al primer paso del trabajo con el almacén: el análisis automático de la estructura relacional existente y la propuesta automática de un modelo dimensional, que pueda ser adaptable a los intereses del usuario.

## Situación Problémica:

El trabajo de creación de Almacenes de Datos en muchas ocasiones se torna un poco complicado puesto que no se poseen herramientas comerciales de análisis y diseño que generen un modelo dimensional adecuado para el almacén, sin un conocimiento previo de los metadatos y modelo relacional de los sistemas de información que se utilizarán como orígenes de los datos a incorporar al almacén.

Teniendo en cuenta esta dificultad antes expresada, se llegó a la conclusión de que se necesita una herramienta CASE que facilite el trabajo de analistas, implementadores y usuarios en cuanto al análisis automatizado de las Bases de Datos relacionales existentes y proponer una estructura estrella óptima para el Almacén, que sea simple de usar para los usuarios aunque estos no tengan todo el conocimiento y experiencia requerido sobre el tema.

Esta herramienta debe permitir el desarrollo visual de diagramas con un modelo estrella o derivado, a partir de estructuras Entidad-Relación de sistemas de información heredados, que tributarán datos al almacén, permitiendo generar la estructura de tablas de hechos y dimensionales, así como los procedimientos de carga del almacén.

## Objetivo General:

Desarrollar una primera versión de una herramienta CASE que apoye el análisis y diseño de sistemas de información para la toma de decisiones (almacenes de datos), preferentemente basado en el modelo dimensional, y que permita generar propuestas de dicho modelo a partir de la revisión automática del modelo relacional de los sistemas de información heredados.

## Objetivos específicos:

1. Estudiar las características propias del modelo dimensional (esquemas estrella o copo de nieve) utilizado mayormente en la confección de Almacenes de Datos y establecer sus diferencias con respecto al Modelo Relacional en que se basan la mayoría de los sistemas de información actuales.
2. Determinar y especificar algunas pautas de conversión de estructuras relacionales a estructuras dimensionales, que permitan establecer un algoritmo de definición de posibles

tablas de hechos y dimensionales, a partir de los metadatos estructurales de los sistemas heredados de determinados formatos relacionales.

3. Implementar una herramienta CASE, al menos en una primera versión, que aplique las pautas de conversión anteriormente especificadas, y presente al usuario propuestas de tablas de hechos y dimensionales, a partir de lo cual, se pueda proponer automáticamente una parte del modelo estrella o copo de nieve del almacén de datos sugerido en base a los requerimientos del usuario.
4. Probar la conversión de parte del modelo relacional de un sistema tomado como caso de estudio, a uno de los esquemas dimensionales solicitados.

## Preguntas de Investigación:

1. ¿Cuáles son las diferencias sustanciales entre el modelo relacional y el modelo dimensional, que implican una transformación engorrosa al crear Almacenes de Datos?
2. ¿Qué pasos o pautas debe tomar en cuenta un analista de sistema para crear la estructura de un almacén de datos usando el modelo estrella a partir del modelo relacional de un sistema de información tradicional?
3. ¿Se podrá implementar una herramienta CASE que detecte a través del análisis de los metadatos de un sistema de información heredado, y que aplique las pautas de conversión que se establezcan, para poder hacer propuestas automáticas adecuadas de partes del modelo dimensional para la implementación del almacén de datos?
4. ¿Podrá ser probado dicho software con varios formatos de Sistemas de Gestión de Bases de Datos?

## Viabilidad de la Investigación:

Una de las temáticas que más desarrollo demanda en los momentos actuales en cuanto a las tecnologías software es sin dudas la Ingeniería del Software, y dentro de esta, el desarrollo de métodos y algoritmos apropiados que conlleven a la creación y explotación de herramientas que asistan por medios computacionales al desarrollo de otros sistemas (herramientas CASE).

Dentro del campo de los Sistemas de Información, basados en la explotación de Sistemas de Bases de Datos, en los últimos tiempos se ha desarrollado la creación y explotación de Almacenes de Datos como uno de los típicos Sistemas de Ayuda a la Toma de Decisiones. Estos sistemas que por concepto manipulan información histórica recopilada en sistemas de información tradicionales de gestión empresarial, con el objetivo de descubrir nuevas informaciones que permitan avanzar en productividad y logros de todo tipo, se caracterizan entre otras cosas -según varios autores entre los que se pueden mencionar a Immon y Kimbal- por tener una estructura de tablas bastante

diferente a la heredada de los sistemas tradicionales (casi siempre relacionales), conocida por Modelo Dimensional o Estrella.

La creación y explotación de Almacenes de Datos pasa entonces por varias fases a saber: Diseño y Creación del Almacén, Proceso de Carga y Limpieza de Datos, Aplicación de Métodos de Descubrimiento de Información (con métodos estadísticos o de minería de datos fundamentalmente), etc.

La primera etapa mencionada requiere hasta ahora del trabajo basado en la experiencia de los analistas de sistemas, pero que parte del estudio del o de los sistemas de información heredados, que ofertarán sus datos al almacén. No obstante hay varias investigaciones incipientes que tienden de una forma o de otra a automatizar de alguna manera esta labor.

El objetivo central de una de las tareas planteadas en el Proyecto de Investigación “Desarrollo de métodos y tecnologías avanzadas de Ingeniería de Software para el desarrollo de Sistemas de Información”, que dirige el tutor de este trabajo, consiste entonces en el estudio y planteamiento de Métodos, Algoritmos, Reglas, etc., que posibiliten analizar automáticamente, las estructuras de los sistemas de información heredados (en el modelo relacional) y logren proponer una nueva estructura (modelo estrella) para el almacén de datos, a partir de la determinación de que elementos (tablas y atributos) del sistema heredado se convertirá en Hechos y que elementos se convertirá en atributos Dimensionales del Almacén a proponer, y que estos permitan desarrollar herramientas CASE que sirvan de ayuda a la creación de almacenes de datos para diferentes tipos de empresas del país, lográndose entonces un avance en la Ingeniería del Software nacional.

Para esto se hace necesario contar con las experiencias y hábitos de diferentes analistas dedicados en la práctica a la creación de almacenes y además con una bibliografía adecuada que aporte ideas sobre el desarrollo de esta etapa en la creación de los almacenes. Estas necesidades se cubren parcialmente por lo que habrá que desarrollar un fuerte trabajo de investigación.

El presente trabajo sería entonces una primera tentativa de solución de esta tarea de investigación.

## Tareas investigativas:

1. Realización de estudios sobre las técnicas y prácticas empleadas en el desarrollo de una herramienta CASE de Análisis y Diseño para Almacenes de Datos, así como de las características de los modelos relacionales y dimensionales de bases de datos.
2. Estudio de posibles tecnologías y herramientas programáticas a utilizar durante el proceso de desarrollo de una herramienta CASE.
3. Validación del software propuesta con un caso de estudio (sistema de información) determinado.

## Idea a defender:

Con el desarrollo de esta herramienta se apoyará a la creación de almacenes de datos para diferentes tipos de empresas del país.

## Estructura de la tesis:

Para lograr una adecuada organización de esta tesis, la misma se ha distribuido en tres capítulos que se refieren a:

- ✓ **Capítulo 1. Fundamentación Teórica**: Marco teórico relativo a los conceptos necesarios para el desarrollo de este trabajo incluyendo Sistemas de Información, Bases de Datos Relacionales, Almacenes de Datos, herramientas CASE, etc.
- ✓ **Capítulo 2. Análisis y Modelado de la Herramienta CASE DWH-Inspector**: Modelado con UML del software desarrollado.
- ✓ **Capítulo 3. Descripción de Características del CASE**: Breve explicación de las características y opciones de la herramienta CASE desarrollada para permitir su explotación.

# Capítulo 1.

# “Fundamentación Teórica”

# CAPÍTULO 1. “FUNDAMENTACIÓN TEÓRICA”

En este capítulo se tratará todo lo relacionado al estado del arte sobre esta temática, se abordará sobre la metodología utilizada, de las aplicaciones que han servido para la implementación de la herramienta, y se exponen las características y los métodos más utilizados de los almacenes de datos, sus cualidades, ventajas, desventajas, así como, las herramientas de consultas y análisis.

El *Data Warehousing*, es el encargado de extraer, transformar, consolidar, integrar y centralizar los datos que la empresa genera en todos los ámbitos de su actividad diaria de negocios (compras, ventas, producción, etc.) o la información externa relacionada. Permite de esta manera el acceso y exploración de la información requerida, a través de una amplia gama de posibilidades de análisis multivariables, con el objetivo final de dar soporte al proceso de toma de decisiones estratégico y táctico.

## 1.1 Características del *Data Warehouse*.

El *Data Warehousing* posibilita la extracción de datos de sistemas operacionales y fuentes externas, permite la integración y homogeneización de los datos de toda la empresa, provee información que ha sido transformada y resumida, para que ayude en el proceso de toma de decisiones estratégicas y tácticas.

El *Data Warehousing*, convertirá entonces los datos operacionales de la empresa en una herramienta competitiva, debido a que pondrá a disposición de los usuarios indicados la información pertinente, correcta e integrada, en el momento que se necesita.

Pero para que el Data Warehousing pueda cumplir con sus objetivos es necesario, que la información que se extrae, se transforma y se consolida, sea almacenada de manera centralizada en una base de datos con estructura multidimensional denominada **Data Warehouse** (DW).

Una de las definiciones más famosas sobre DW, es la de William Harvey Inmon, quien define: “*Un Data Warehouse es una colección de datos orientada al negocio, integrada, variante en el tiempo y no volátil para el soporte del proceso de toma de decisiones de la gerencia*”(Immon 1996).

Debido a que W. H. Inmon, es reconocido mundialmente como el padre del DW, la explicación de las características más sobresalientes de esta herramienta se basaron en su definición.

Pero antes de pasar al siguiente epígrafe también debemos destacar dos conceptos de Kimball con respecto al modelo a utilizar: “Los modelos *entidad – interrelación*, son un desastre para las consultas debido a que ellos no pueden ser comprendidos por los usuarios y ellos no pueden ser útilmente recorridos por el software del SGBD”. “*Los modelos entidad – interrelación, **no** pueden ser usados como bases para el almacén de datos empresarial”(Kimball 1998).*

Inmon defiende una metodología descendente (*top-down*) a la hora de diseñar un almacén de datos, ya que de esta forma se considerarán mejor todos los datos corporativos.

Ralph Kimball por su parte define los *Data Warehouse* como: "una copia de las transacciones de datos específicamente estructurada para la consulta y el análisis". También fue Kimball quien determinó que un DW no era más que: "la unión de todos los *Data Marts* (Mercados de Datos: versión especial de almacén de datos) de una entidad". Defiende por tanto una metodología ascendente (*bottom-up*) a la hora de diseñar un almacén de datos.

¿Qué es un Data Warehouse según Ralph Kimball (1998)?

- ✓ Es una fuente de datos de la empresa que puede ser consultada.
- ✓ No debe ser organizada con ayuda del modelo entidad/interrelación.
- ✓ Es frecuentemente modificada, a partir de datos correctos.

Las definiciones anteriores se centran en los datos en sí mismos. Sin embargo, los medios para obtener y analizar esos datos, para extraerlos, transformarlos y cargarlos, así como las diferentes formas para realizar la gestión de datos son componentes esenciales de un almacén de datos.

Muchas referencias a un almacén de datos utilizan esta definición más amplia. Por lo tanto, en esta definición se incluyen herramientas para la inteligencia empresarial, herramientas para extraer, transformar y cargar datos (ETL) en el almacén de datos, y herramientas para gestionar y recuperar los metadatos.

Varias han sido las definiciones acerca del término de un Data Warehouse:

Bill Inmon, fue uno de los primeros autores en escribir sobre el tema en términos de las características del almacén de datos:

“Un almacén de datos es una colección de datos

- ✓ ***orientada a temas o materias,***
- ✓ ***integrada,***
- ✓ ***variable en el tiempo y***
- ✓ ***no volátil***

que se usa para el soporte del proceso de toma de decisiones gerenciales. ”

## Orientado a temas

Los datos en la base de datos están organizados de manera que todos los elementos de datos relativos al mismo evento u objeto del mundo real queden unidos entre sí.

Esta definición del DW clasifica la información en base a los aspectos que son de interés para la empresa. Dicha clasificación afecta el diseño y la implementación de los datos encontrados en el

almacén de datos, debido a que la estructura del mismo difiere considerablemente a la de los clásicos procesos operacionales orientados a las aplicaciones.

En síntesis, la ventaja de contar con procesos orientados a la aplicación, está fundamentada en la alta accesibilidad de los datos, lo que implica un elevado desempeño y velocidad en la ejecución de consultas, ya que las mismas están predeterminadas; mientras que en el DW para satisfacer esta ventaja se requiere que la información no esté normalizada, es decir, con redundancia, duplicidad de los datos y que la misma esté dimensionada, para evitar tener que recorrer toda la base de datos cuando se necesite realizar algún análisis determinado, sino que simplemente la consulta sea enfocada por vectores y variables que permitan localizar los datos de manera rápida y eficaz, para poder de esta manera satisfacer una alta demanda de complejos exámenes en un mínimo tiempo de respuesta.(Bernabue 2007)

## Integrado

La base de datos contiene los datos de todos los sistemas operacionales de la organización, y dichos datos deben ser consistentes.

La integración implica que todos los datos de diversas fuentes que son producidos por distintos departamentos, secciones y aplicaciones, tanto internas como externas, deben ser consolidados en una instancia antes de ser agregados al DW. A este proceso se lo conoce como Extracción, Transformación y Carga de Datos (ETL). (Bernabue 2007)

## Variante en el tiempo

Los cambios producidos en los datos a lo largo del tiempo quedan registrados para que los informes que se puedan generar reflejen esas variaciones.

Debido al gran volumen de información que se manejará en el DW, cuando se le realiza una consulta, los resultados deseados demorarán en originarse. Este espacio de tiempo que se produce desde la búsqueda de datos hasta su consecución es del todo normal en este ambiente y es, precisamente por ello, que la información que se encuentra dentro del depósito de datos se denomina de tiempo variable.

El intervalo de tiempo y periodicidad de los datos debe definirse de acuerdo a la necesidad y requisitos de los usuarios.

Es elemental aclarar, que el almacenamiento de datos históricos, es lo que permite al DW desarrollar pronósticos y análisis de tendencias y patrones, a partir de una base estadística de información, ya que las instantáneas son actualizadas de acuerdo con las actividades del negocio.(Bernabue 2007)

**No volátil**

La información no se modifica ni se elimina, una vez almacenado un dato, éste se convierte en información de sólo lectura, y se mantiene para futuras consultas.

La información es útil para el análisis y la toma de decisiones solo cuando es estable. Los datos operacionales varían momento a momento, en cambio, los datos una vez que entran en el DW no cambian.

La actualización, o sea, insertar, eliminar y modificar, se hace de forma muy habitual en el ambiente operacional sobre una base, registro por registro, en cambio en el depósito de datos la manipulación básica de los datos es mucho más simple, debido a que solo existen dos tipos de operaciones: la carga de datos y el acceso a los mismos.

Por esta razón es que en el DW no se requieren mecanismos de control de la concurrencia y recuperación.(Bernabue 2007)

## 1.2. Métodos más usados en la Construcción del *Data Warehouse*

Los modelos propuestos por William H. Inmon y Ralph Kimball para llevar a cabo el diseño de un Data Warehouse son los más aplicados en la actualidad, coincidiendo en que un *Data Mart* o un mercado de datos independiente no satisface las necesidades que tienen las compañías a escala corporativa de acceder inmediatamente y con facilidad a sus datos, pero sus criterios difieren en cuanto al modelo de datos y a las arquitecturas.

El término *Data Mart* es usado para designar a los almacenes de datos cuyo ámbito es más reducido, normalmente un departamento o área específica dentro de la empresa, es definido por Ralph Kimball como bodegas de datos con información de interés particular para un determinado sector de la empresa y aunque su enfoque sea para una sola perspectiva departamental, no lo exime de tener que seguir los lineamientos generales de implementación que posee el *Data Warehouse* (KIMBALL 1996).

Ralph Kimball propone como modelo de datos al modelo dimensional, el más popular en las soluciones que se implementan de manera práctica, el cual facilita a los usuarios finales las consultas y el análisis. Se caracteriza por ser sencillo de crear, extremadamente estable en presencia de cambios, además de mostrarse muy intuitivo y comprensible; el autor sugiere el uso de este modelo de datos para el desarrollo de los *Data Marts* y del *Data Warehouse* (KIMBALL and CASERTA 2004).

También William H. Inmon reconoce al modelo dimensional como el mejor para el desarrollo de los *Data Marts* por las ventajas brindadas, pero propone la construcción del *Data Warehouse* basado en el modelo entidad-relación. La idea de Inmon se basa en que el modelo entidad relación es mucho más rico y adaptable que el dimensional (INMON 2002).

En cuanto a la arquitectura William H. Inmon en su libro "*Building the Data Warehouse*" (INMON, 2005) plantea que la construcción del *Data Warehouse* no debe ser sustituida por la implementación de varios *Data Marts*.

Resaltando que la excusa para no desarrollar un almacén de datos la mayoría de las veces es por no contar con un gran presupuesto, la sustitución de este por los *Data Marts* trae desventajas puesto que están diseñados para un área particular de la empresa, lo que trae consigo diferencias entre las estructuras de datos de los mismos, que al integrarlos en el *Data Warehouse* algunos no serán reutilizables, ni flexibles, ni útiles para la reconciliación que se necesita. Inmon manifiesta que el proceso de construcción del *Data Warehouse* parte de los sistemas operacionales existentes, creándose áreas de diferentes temas, cuando existan una cierta cantidad de estas, el *Data Warehouse* inicia el proceso de población de las áreas de una manera integrada, una vez concluido se comienza a dar respuestas a las inquietudes de los usuarios; empezando así el florecimiento del nivel departamental a medida que se tienen más datos en el *Data Warehouse* y es en este punto del desarrollo cuando se centra la atención en las cuestiones de los diferentes departamentos, para definir y crear los almacenes de datos departamentales, los *Data Marts*.

Ralph Kimball en desacuerdo con la arquitectura propuesta por William H. Inmon resalta en su libro "*The Data Warehouse ETL Toolkit, Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*" (KIMBALL and CASERTA 2004), que los *Data Marts* están basados en los datos de la fuente y no en la visión departamental, en otras palabras que el *Data Mart* es sólo una parte de un producto orientado a la compañía, los cuales deben consistir en una continua pirámide de estructuras dimensionales idénticas, comenzando siempre con los datos atómicos. Plantea también que la idea de construcción de un *Data Warehouse* centralizado no es realista, siendo más real construirlo en un ambiente descentralizado e incremental, porque las empresas están en constante cambio, adquiriendo nuevas fuentes de datos y necesitando nuevas perspectivas, propone además, centrarse en trazar estrategias adaptables e incrementales basándose en una idealista visión de controlar toda la información antes de construir el *Data Warehouse*. Por esta razón manifiesta que el proceso de construcción de un almacén de datos parte de los sistemas operacionales existentes, creando los diferentes *Data Marts* basados en la información de dichas fuentes, para luego de tenerlos desarrollados y funcionales se comience con la construcción del *Data Warehouse* basado en la información que éstos contienen.

En la actualidad este método es el más usado, gracias a las diferentes ventajas que proporciona, permitiendo a las empresas acometer los proyectos de manera separada y de esta forma reducir los efectos negativos que tendría fracasar en un intento por construir un *Data Warehouse*.

Por tanto en nuestro trabajo abordaremos este enfoque y se seleccionará como problema práctico a considerar un área determinada del proceso de gestión de información de una

Universidad, en específico lo referente a la información que emana del sistema de control de postgrado. Esto puede ser considerado como un Mercado de Datos de una gran institución como es una Universidad. Además el sistema operacional de control de esta actividad fue desarrollado e implantado en nuestro centro por parte del tutor, lo que nos ayuda a determinar las necesidades correspondientes para este trabajo.

## 1.3. Cualidades del Data Warehouse

Una de las primeras cualidades que se puede mencionar del DW, es que maneja un gran volumen de datos, debido a que consolida en su estructura la información recolectada durante años, proveniente de diversas fuentes, en un solo lugar centralizado. Es por esta razón que el depósito puede ser soportado y mantenido sobre diversos medios de almacenamiento.

Además, el almacén de datos presenta la información resumida y agregada desde múltiples versiones, y maneja información histórica. Organiza y almacena los datos que se necesitan para el procesamiento analítico e informático, con el propósito de responder a preguntas de negocios y brindarles a los usuarios finales una interfaz amigable, comprensible y fácil de utilizar, para que los mismos puedan tomar decisiones sobre los datos sin tener que poseer demasiados conocimientos informáticos. El DW permite un acceso más directo, es decir, la información gira en torno al negocio, y es por ello que también los usuarios pueden sentirse cómodos al explorar los datos y encontrar relaciones complejas entre los mismos.

El DW no es solo datos, sino un conjunto de herramientas para consultar, analizar y presentar información, que permiten obtener o realizar análisis, reportes, extracción y explotación de los datos, con alta actuación, para transformar dichos datos en información valiosa para la organización. Sin embargo en lo referente a este trabajo nuestro centro de atención está precisamente en la estructura donde deben almacenarse los datos del almacén que después serán tratados por otras herramientas y con diversos fines.

Con respecto a las tecnologías empleadas, en un almacén de datos se pueden encontrar las siguientes:

- Arquitectura cliente/servidor.
- Técnicas avanzadas para replicar, refrescar y actualizar datos.
- Software *front-end*, para acceso y análisis de datos.
- Herramientas para extraer, transformar y cargar datos en el depósito, desde múltiples fuentes muy heterogéneas.
- Sistema de Gestión de Base de Datos (SGBD).

Cabe destacar, que todas las cualidades expuestas anteriormente, son imposibles de saldar en un típico ambiente operacional, y esto es una de las razones de ser del DW.(Bernabue 2007)

## Ventajas del Data Warehouse

- Transforma datos orientados a las aplicaciones en información orientada a la toma de decisiones.
- Integra y consolida diferentes fuentes de datos y departamentos empresariales, que anteriormente formaban islas, en una única plataforma sólida y centralizada.
- Provee la capacidad de analizar y explotar las diferentes áreas de trabajo y de realizar un análisis inmediato de las mismas.
- Permite reaccionar rápidamente a los cambios del mercado.
- Aumenta la competitividad en el mercado.
- Elimina la producción y el procesamiento de datos que no son utilizados ni necesarios, producto de aplicaciones mal diseñadas o ya no utilizadas.
- Mejora la entrega de información, es decir, información completa, correcta, consistente, oportuna y accesible. Información que los usuarios necesitan, en el momento adecuado y en el formato apropiado.
- Logra un impacto positivo sobre los procesos empresariales. Cuando los usuarios tienen acceso a una mejor calidad de información, la empresa puede lograr por sí misma: aprovechar el enorme valor potencial de sus recursos de información y transformarlo en valor verdadero; eliminar los retardos de los procesos empresariales que resultan de información incorrecta, inconsistente y/o inexistente; integrar y optimizar procesos a través del uso compartido e integrado de las fuentes de información; permitir al usuario adquirir mayor confianza acerca de sus propias decisiones y de las del resto, y lograr así, un mayor entendimiento de los impactos ocasionados.
- Aumento de la competitividad de los encargados de tomar decisiones.
- Los usuarios pueden acceder directamente a la información en línea, lo que contribuye a su capacidad para operar con mayor efectividad en las tareas rutinarias o no. Además, pueden tener a su disposición una gran cantidad de valiosa información multidimensional, presentada coherentemente como fuente única, confiable y disponible en sus estaciones de trabajo. Así mismo, los usuarios tienen la facilidad de contar con herramientas que les son familiares para manipular y evaluar la información obtenida en el DW, tales como: hojas de cálculo, procesadores de texto, software de análisis de datos, software de análisis estadístico, reportes, etc.
- Permite la toma de decisiones estratégicas y tácticas.(Bernabue 2007)

## Desventajas del Data Warehouse

- ✓ Requiere una gran inversión, debido a que su correcta construcción no es tarea sencilla y consume muchos recursos, además, su misma implementación implica desde la adquisición de herramientas de consulta y análisis, hasta la capacitación de los usuarios.

- ✓ Existe resistencia al cambio por parte de los usuarios.
- ✓ Los beneficios del almacén de datos son apreciados en el mediano y largo plazo. Este punto deriva del anterior, y básicamente se refiere a que no todos los usuarios confiarán en el DW en una primera instancia, pero sí lo harán una vez que comprueben su efectividad y ventajas.
- ✓ Además, su correcta utilización surge de la propia experiencia.
- ✓ Si se incluyen datos propios y confidenciales de clientes, proveedores, etc., el depósito de datos atentará contra la privacidad de los mismos, ya que cualquier usuario podrá tener acceso a ellos.
- ✓ Infravaloración de los recursos necesarios para la captura, carga y almacenamiento de los datos.
- ✓ Infravaloración del esfuerzo necesario para su diseño y creación.
- ✓ Incremento continuo de los requerimientos del usuario.(Bernabue 2007)

El almacenamiento de los datos no debe usarse con datos de uso actual. Los almacenes de datos contienen a menudo grandes cantidades de información que se subdividen a veces en unidades lógicas más pequeñas dependiendo del subsistema de la entidad del que procedan o para el que sea necesario. Los *Data Warehouse* surgen con la promesa del manejo y control de la información. Ellos aseguran una vista única de los datos, que pueden provenir de diversas fuentes. Gracias a esto, los usuarios finales no se ven en la necesidad de aprender y utilizar múltiples sistemas de acceso y manipulación de los datos. Un almacén de datos facilita la comprensión de los datos, transformándolos en información útil, teniendo como bandera el apoyo a la Toma de Decisiones. Permite no solo comprender lo que está pasando, sino predecir lo que va a suceder.

El objetivo del *Data Warehouse* es integrar datos corporativos, residentes en bases de datos operacionales de la organización, en un único repositorio sobre el cual los usuarios puedan realizar consultas o informes y hacer análisis de datos.

La tecnología de almacenes de datos integra las técnicas de bases de datos y las técnicas de análisis de datos.

Las bases de datos multidimensionales implican tres variantes posibles de modelamiento, que permiten realizar consultas de soporte de decisión:

## 1.4. Variantes de Modelación de Almacenes de Datos.

- ❖ Esquema en estrella (*Star Scheme*).
- ❖ Esquema copo de nieve (*Snowflake Scheme*).
- ❖ Esquema constelación o copo de estrellas (*Starflake Scheme*).

Estos esquemas pueden ser implementados de diversas maneras, que, independientemente al tipo de arquitectura, requieren que toda la estructura de datos este desnormalizada o semi-

desnormalizada, para evitar desarrollar uniones complejas para acceder a la información, con el fin de agilizar la ejecución de consultas. Los diferentes tipos de implementación son los siguientes:

- Relacional – ROLAP.
- Multidimensional – MOLAP.
- Híbrido – HOLAP.

## Esquema en Estrella

El esquema en estrella, consta de una tabla de hechos central y de varias tablas de dimensiones relacionadas a esta, a través de sus respectivas claves, es un paradigma de modelado que tiene un solo objeto en el medio conectado con varios objetos de manera radial. Refleja la visión del usuario final de una consulta empresarial. Un esquema estrella lógico sencillo tiene una sola tabla de hechos y varias dimensiones conectadas a ella. En la figura que sigue se muestra un esquema estrella estándar:

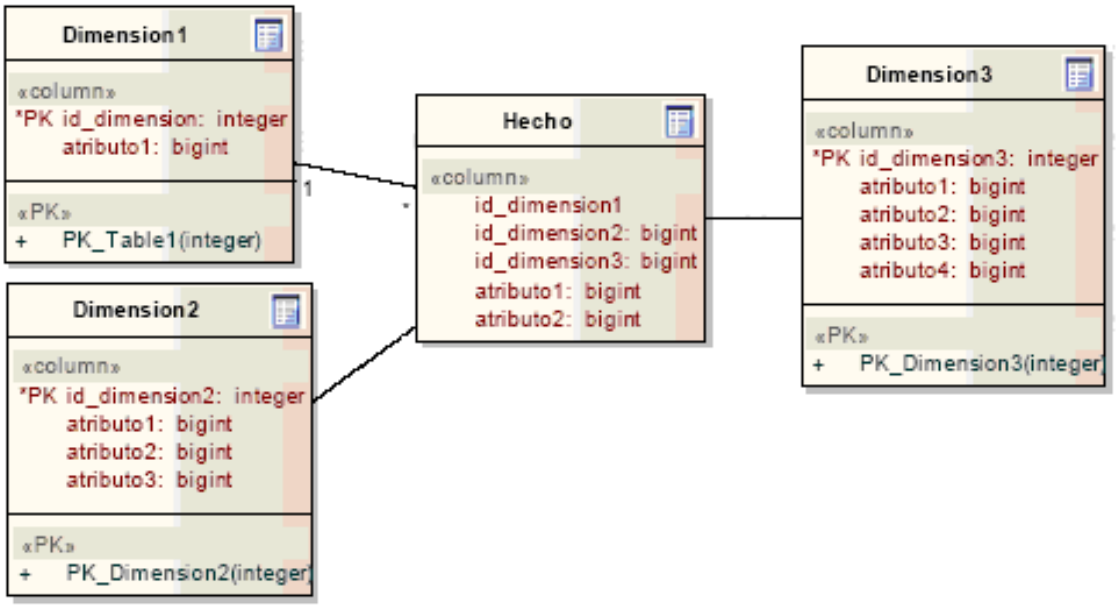


Figura 1. Esquema Estrella estándar

### *Características del Esquema en Estrella*

- ✓ Posee los mejores tiempos de respuesta.
- ✓ Su diseño es fácilmente modificable.
- ✓ Existe paralelismo entre su diseño y la forma en que los usuarios visualizan y manipulan los datos.
- ✓ Simplifica el análisis.
- ✓ Facilita la interacción con herramientas de consulta y análisis.(Bernabue 2007)

## Esquema Copo de Nieve

Este esquema representa una extensión del modelo en estrella cuando las dimensiones se organizan en jerarquías de dimensiones.(Bernabue 2007)

Es una extensión del esquema estrella en donde cada uno de los puntos de la estrella se divide en más puntos y tiene como ventaja mejor desempeño en consultas, aplica normalización y con ello más flexibilidad de las aplicaciones. En la figura que sigue se puede apreciar un esquema de Copo de Nieve estándar:

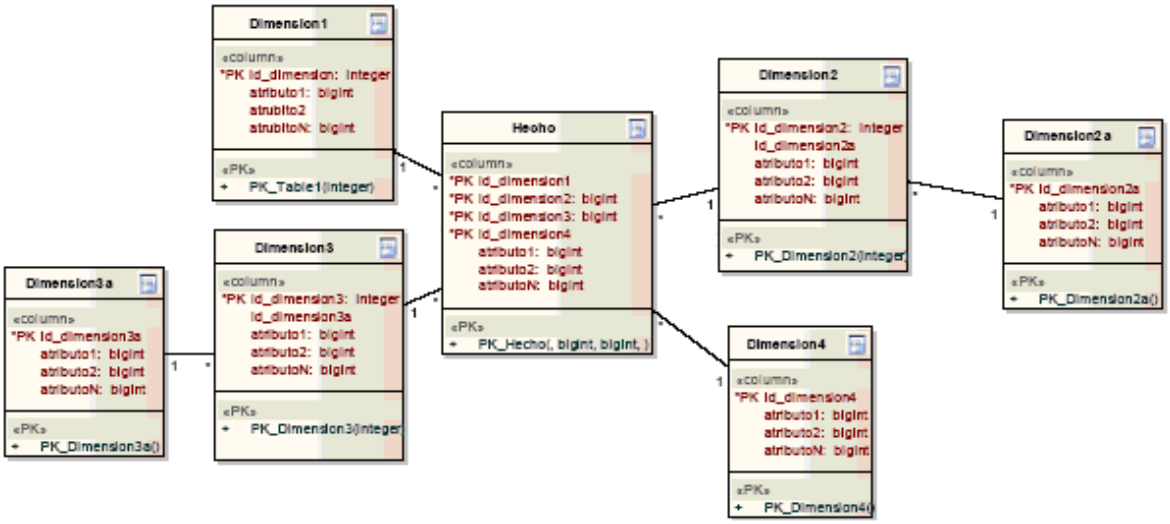


Figura 2. Esquema Copo de Nieve estándar

### *Se pueden definir las siguientes características de este tipo de modelo:*

- Posee mayor complejidad en su estructura.
- Hace una mejor utilización del espacio.
- Es muy útil en tablas de dimensiones de muchas tuplas.
- Las dimensiones están normalizadas, por lo que requiere menos esfuerzo de diseño.
- Puede desarrollar clases de jerarquías fuera de las dimensiones, que permiten realizar análisis de lo general a lo detallado y viceversa.

A pesar de todas las características y ventajas que trae aparejada la implementación del esquema copo de nieve, existen dos grandes inconvenientes de ello:

- Si se poseen múltiples dimensiones, cada una de ellas con varias jerarquías, se creará un número de dimensiones bastante considerable, que pueden llegar al punto de ser inmanejables.
- Al existir muchas uniones y relaciones entre tablas, el desempeño puede verse reducido.

## Esquema Constelación

Este modelo está compuesto por una serie de esquemas en estrella, y está formado por una tabla de hechos principal y por una o más tablas de hechos auxiliares, las cuales pueden ser resúmenes de la principal. Dichas tablas yacen en el centro del modelo y están relacionadas con sus respectivas tablas de dimensiones.

No es necesario que las diferentes tablas de hechos compartan las mismas tablas de dimensiones, ya que, las tablas de hechos auxiliares pueden vincularse con solo algunas de las tablas de dimensiones asignadas a la tabla de hechos principal, y también pueden hacerlo con nuevas tablas de dimensiones.

Su diseño y cualidad es muy similar a la del esquema en estrella, pero posee una serie de diferencias con el mismo. Entre ellas se pueden mencionar:

- Permite tener más de una tabla de hechos, por lo cual se podrán analizar más aspectos claves del negocio con un mínimo esfuerzo adicional de diseño.
- Contribuye a la reutilización de dimensiones, ya que una misma dimensión puede utilizarse para varias tablas de hechos.
- No es soportado por todas las herramientas de consulta y análisis.(Bernabue 2007)

En la figura que sigue se puede apreciar un esquema de Constelación estándar:

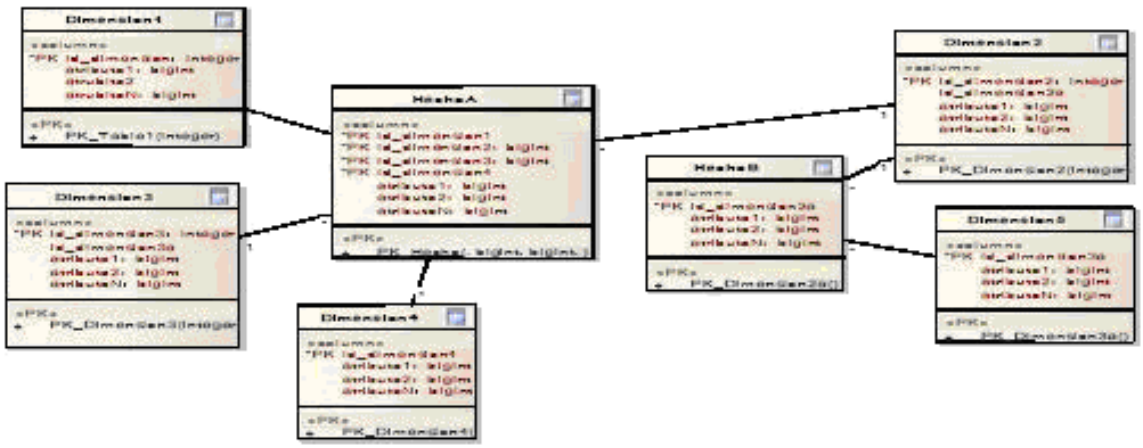

Figura 3. Esquema Constelación estándar

El esquema estrella, contiene una estructura de depósito de datos que se adapta mejor a los requerimientos y necesidades del usuario. Al contar con baja cantidad de dimensiones ofrece respuestas más rápidas a preguntas más complejas. Por lo hemos decidido basarnos en el mismo para el desarrollo de este trabajo. No obstante es de señalar que hay problemas reales donde varias tablas de hechos pueden encadenarse para dar una idea global de una cadena de gestión de información y esto se acomoda al esquema de constelación.

El modelo multidimensional dentro del entorno de las bases de datos, es una disciplina de diseño que se sustenta en el modelo entidad relación y en las realidades de la ingeniería de texto y datos numéricos.(Kimball 1995)

Dadas las características de los almacenes de datos es ideal la utilización en su diseño de un Modelo Multidimensional (MMD). Este tipo de diseño tiene como ventajas sobre el Modelo Entidad-Relación (MER), que es muy flexible, está desnormalizado y orientado a los intereses de un usuario final, aunque esto no significa que existan inconsistencias en los datos. Mediante la utilización de un MMD se disminuye la cantidad de tablas y relaciones entre ellas, lo que agiliza el acceso a los datos.(Kimball 1995)

El modelo multidimensional se representa a través de la definición de las tablas de hechos y dimensiones.

## *1.5. El Modelo Dimensional.*

Una dimensión representa una perspectiva de los datos. Las dimensiones son usadas para seleccionar y agregar datos a un cierto nivel deseado de detalle. Podemos definir el concepto de dimensión como el grado de libertad de movimiento en el espacio. Entenderemos esta libertad como el número de direcciones ortogonales diferentes que podamos tomar.

Las dimensiones se relacionan en jerarquías o niveles. Por ejemplo, la dimensión Zona puede tener los siguientes niveles: ciudad, estado, región, país y continente.(Anónimo)

Las tablas de dimensiones definen como están los datos organizados lógicamente y proveen el medio para analizar el contexto del negocio. Representan los ejes del cubo, y los aspectos de interés, mediante los cuales el usuario podrá filtrar y manipular la información almacenada en la tabla de hechos.(Bernabue 2007)

Los talleres de sistemas de información, los usuarios finales, y aún las asociaciones de suministradores de datos como A. C. Nielsen, IRI, IMS, y Walsh América han gravitado caso por caso a una simple estructura de “cubo de datos” que se equipara a las necesidades de los usuarios finales por simplicidad. Esta estructura es en definitiva un modelo dimensional. Un Modelo Dimensional típico aparece en la figura 4.

Otro nombre para el modelo dimensional es el de esquema de enlace estrella debido a que los esquemas se parecen a una estrella, con una gran tabla central y un conjunto de pequeñas tablas acompañantes presentadas en un modelo radial alrededor de la tabla central.

A diferencia del modelo entidad –interrelación el modelo dimensional es muy asimétrico.

- ✓ Existe aquí una gran tabla dominante en el centro del esquema.
- ✓ Ella es la única tabla en el esquema con múltiples enlaces conectándola a otras tablas.
- ✓ Las otras tablas tienen todas, un enlace simple que las enlaza con la tabla central.

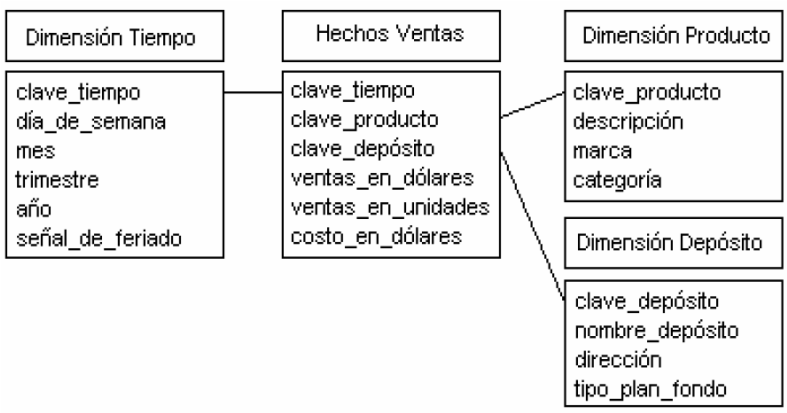


Figura 4. Modelo Dimensional típico

Las tablas dimensionales por su parte son aquellas donde las descripciones textuales de las dimensiones del negocio son almacenadas. Cada una de las descripciones textuales ayuda a describir un miembro de la dimensión respectiva. En una base de datos bien diseñada, la tabla dimensional producto tendrá muchos atributos (campos). Sin embargo, en los ejemplos desarrollados por varios autores como Kimball, resalta la existencia en casi todos los Almacenes de datos de las Dimensiones Tiempo y Localización (geográfica).

Las tablas de hechos contienen los hechos, medidas o indicadores que serán utilizados por los analistas de negocio para apoyar el proceso de toma de decisiones. Los hechos son datos instantáneos en el tiempo, que son filtrados, agrupados y explorados a través de condiciones definidas en las tablas de dimensiones.(Bernabue 2007)

En la tabla de hechos es donde las mediciones numéricas del negocio son almacenadas. Cada una de las mediciones es tomada como la intersección de todas las dimensiones, los mejores y más útiles hechos son numéricos, valorados continuamente y aditivos.

La razón para los hechos numéricos, valorados continuamente y aditivos, es que en cada consulta hecha contra esa tabla de hechos estamos preguntando por cientos, miles o millones de registros a ser usados por el SGBD para construir el conjunto respuesta.

Este gran número de registros será comprimido a unas pocas docenas de filas del conjunto respuesta del usuario. La única forma útil de compactar estos registros en el conjunto respuesta es adicionándolos. Por tanto, si las mediciones son numéricas y si ellas son aditivas, podemos construir fácilmente el conjunto respuesta.

En otros casos hay hechos que son semiaditivos, que pueden ser adicionados a lo largo de solo algunas dimensiones. También hay hechos que son no aditivos, los que simplemente no pueden adicionarse del todo. Para los hechos no aditivos, estamos obligados a usar contadores si se desea resumir los registros.

## 1.6. El Modelo Relacional, los SGBDR, el Modelado Conceptual Entidad-Relación y los Sistemas de Información.

El modelo relacional fue propuesto en primer lugar por E. F. Codd en su artículo original titulado '*A relational model of data for large shared data banks*' (CODD, 1970). Este artículo se acepta ahora generalmente como uno de los hitos principales en los sistemas de bases de datos, aunque anteriormente se había propuesto un modelo orientado a conjuntos (CHILDS, 1968). Los objetivos especificados del modelo relacional eran:

- ✓ Permitir un alto grado de independencia de los datos. Los programas de aplicación no deben verse afectados por las modificaciones efectuadas en la representación interna de los datos, y en particular por los cambios efectuados en la organización de los archivos, en la ordenación de los registros o en las rutas de acceso.
- ✓ Proporcionar una base teórica sólida que permitiera tratar con la semántica de los datos y con los problemas de coherencia y de redundancia. En particular, el artículo de Codd introducía el concepto de relaciones normalizadas, es decir, relaciones en las que no haya grupos repetidos (el proceso de normalización incluye cinco formas normales, que no es objetivo explicar acá).
- ✓ Permitir la ampliación de lenguajes de manipulación de datos orientados a conjuntos.

Aunque el interés en el modelo relacional se manifestó desde múltiples direcciones, la investigación más significativa puede atribuirse a tres proyectos que tenían perspectivas bastante distintas. El primero de éstos, en el laboratorio *San Jose Research Laboratory* de IBM, en California, fue el SGBD relacional prototipo *System R*, que fue desarrollado a finales de la década de 1970. Este proyecto se llevó a cabo para demostrar la posibilidad de implementar el modelo relacional, desarrollando un ejemplo de sus estructuras de datos de operaciones. También demostró ser una fuente excelente de información acerca de problemas de implementación tales como la gestión de transacciones, el control de concurrencia, las técnicas de recuperación, la optimización de consultas, los problemas de seguridad e integridad de los datos, los factores humanos y las interfaces de usuario, y condujo a la publicación de muchos artículos de investigación en el desarrollo de otros prototipos. En particular, el proyecto *System R* condujo a dos desarrollos principales:

- ✓ el desarrollo de un lenguaje de consulta estructurado denominado SQL, que desde entonces se ha convertido en e] lenguaje estándar formal de ISO (*International Organization for Standardization*) y en el lenguaje estándar de facto para los SGBD relacionales;

- ✓ el desarrollo de varios productos comerciales de SGBD relacional a finales de la década de 1970 y principios de la de 1980, como por ejemplo DB2 y SQL/DS de IBM y Oracle de Oracle Corporation.

El segundo proyecto de importancia en el desarrollo del modelo relacional fue INGRES (*Interactive Graphics Retrieval System*, sistema gráfico interactivo de extracción) en la Universidad de California en Berkeley, que más o menos se desarrolló al mismo tiempo que el proyecto *System R*. El proyecto INGRES implicaba el desarrollo de un prototipo de SGBDR, concentrándose en la investigación en los mismos objetivos globales que el proyecto *System R*. Estas investigaciones condujeron a una versión académica de INGRES que contribuyó a la popularización de los conceptos relacionales y dio como resultado los productos comerciales INGRES de *Relational Technology Inc.* (ahora *Advantage Ingres Enterprise Relational Database* de *Computer Associates*) e *Intelligent Database Machine* de *Britton Lee lnc.*

El tercer proyecto fue el denominado *Peterlee Relational Test Vehicle* en el laboratorio del IBM *Scientific Centre en Peterlee*, Reino Unido (Todd, 1976). Este proyecto tenía una orientación más teórica que los proyectos *System R* e INGRES y su principal importancia radica en la investigación en cuestiones tales como el procesamiento y optimización de consultas y la ampliación funcional.

Los sistemas comerciales basados en el modelo relacional comenzaron a aparecer a finales de la década de 1970 y principios de la de 1980. Ahora hay centenares de SGBDR para entornas tanto *mainframe* como PC, aunque muchos de ellos no se adhieren estrictamente a la definición del modelo relacional. Como ejemplos de SGBDR basado en PC podemos citar *Office Access* y *Visual FoxPro* de *Microsoft*, *InterBase* y *JDataStore* de *Borland* y *R:Base* de *R:BASE Technologies*.

El modelo relacional está basado en el concepto matemático de relación, la cual se representa físicamente en forma de una tabla. Codd, que tenía formación matemática, utilizó terminología sacada del campo de las matemáticas, principalmente de la teoría de conjuntos y de la lógica de predicados. Un resumen de la terminología utilizada por el modelo relacional es:

- ✓ **Relación**: Una relación es una tabla con columnas y filas.
- ✓ **Atributo**: Un atributo es una columna nominada de una relación.
- ✓ **Dominio**: Un dominio es un conjunto de valores permitidos para uno o más atributos.
- ✓ **Tupla**: Una tupla es una fila de una relación.
- ✓ **Grado**: El grado de una relación es el número de atributos que contiene.
- ✓ **Cardinalidad**: La cardinalidad de una relación es el número de tuplas que contiene.
- ✓ **Base de datos relacional**: Una colección de relaciones normalizadas en la que cada relación tiene un nombre distintivo.

El uso de un SGBD facilita las características de control de redundancia, almacenamiento persistente de las características de los objetos y estructuras de datos, suministro de múltiples interfaces con los usuarios, cumplimiento de restricciones de integridad, así como también menor tiempo de creación de aplicaciones, flexibilidad y disponibilidad de información actualizada.

Cada SGBD posee intrínsecamente un lenguaje de definición de datos (LDD) y un lenguaje de manipulación de datos (LMD), lo que favorece el desarrollo de aplicaciones sin necesidad de complicadas estructuras de programación. Por otra parte con el desarrollo de sistemas de información basados en SGBD es posible implementar adecuadas interfaces de usuario basadas en menúes, formularios e informes principalmente, lo cual es una tendencia actual.

Los SGBDR modernos ofrecen entre otras facilidades, la de contar con un SQL empotrado, lo cual no es más que un conjunto de instrucciones que permite que el lenguaje estructurado de consultas sea utilizado con lenguajes de programación imperativos. A su vez, casi todos estos gestores incluyen dos variantes de creación de las consultas: a través del SQL directamente o a través de módulos de definición de consultas, basado en un lenguaje no procedimental, conocidos como QBE (*Query By Example*, consultas por ejemplo).

El Modelo Entidad - Interrelación (ER) fue introducido por Chen en 1976. En este modelo la información es representada por medio de tres conceptos primitivos:

- ✓ **Entidades**, las cuales representan los objetos a ser modelados.
- ✓ **Atributos**, los cuales representan las propiedades de dichos objetos.
- ✓ **Interrelaciones**, las cuales representan los vínculos entre entidades.

El Modelo Entidad-Interrelación es un modelo conceptual de alto nivel, usado frecuentemente para el diseño de aplicaciones de bases de datos.

El esquema ER correspondiente al diseño de la estructura de una base de datos es presentado de forma gráfica, donde las entidades se dibujan en forma de rectángulos, los atributos se dibujan en forma de elipses, y las interrelaciones se dibujan en forma de rombos.

En cualquier libro de texto sobre bases de datos y en varios artículos, se explica claramente este modelo semántico de diseño de bases de datos. En el desarrollo de un sistema de información para aceros, por ejemplo se deben definir como entidades entre otras las siguientes:

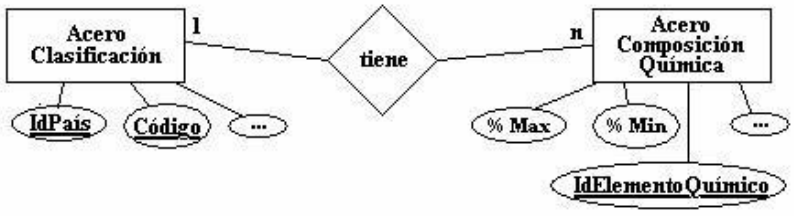


Figura 5. Modelo Entidad-Relación típico

Un Sistema de Información puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar la toma de decisiones, la coordinación y el control en una institución.

Existen diferentes tipos de sistemas de información pero los más característicos para este trabajo son los sistemas de gestión de la información básica de las empresas generalmente clasificados como temporales pues manipulan información relativa a ventas, producciones, costos, etc., para un período de tiempo determinado; y los sistemas de apoyo a la toma de decisiones, que se basan sobre todo en datos permanentes que permiten hacer valoraciones especiales. Estos últimos se desarrollan actualmente mediante las técnicas de almacenes de datos.

## 1.7. Herramientas CASE específicas para Bases de Datos.

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante algunos pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación, Implantación y Mantenimiento.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad. Para conseguir estos dos objetivos es conveniente contar con una organización y una metodología de trabajo, además de la propia herramienta.

No fue sino hasta 1985 en que las herramientas CASE se volvieron realmente importantes en el proceso de desarrollo de software. Los proveedores prometieron a la Industria que muchas actividades serían beneficiadas por la ayuda de las CASE. Estos beneficios consistían, por ejemplo, en el aumento en la productividad. El objetivo en 1985 para muchos vendedores era producir software más rápidamente. Las herramientas del CASE serían una familia de métodos favorablemente estructurados para planeamiento, análisis y diseño. Esto llevaría a la generación automática de código para desarrollo de software vía una especificación formalmente diseñada. Esto traería como beneficio:

1. Una mejora en la calidad, fiabilidad, utilidad y rendimiento.
2. El entorno de producción de documentación para comunicación, mantenimiento y actualización.
3. Hace el trabajo de diseño de software más fácil y agradable.
4. La promesa futura de reemplazar realmente a los ingenieros de software especializados.
5. Reducción del costo de producción de software.(Anónimo 1996)

Aunque existen varias herramientas CASE, algunas de las cuales servían para guiar el proceso de desarrollo de software con el uso del paradigma estructurado, una gran mayoría actual se basa

en la ayuda al desarrollo de software orientado a objetos. Son característicos los casos de CASE como *Rational Rose*, *Visual Paradigm*, *Visual UML*, *Ptech*, etc. No obstante, siendo este trabajo específico para la modelación y generación automática de esquemas de bases de datos, se hará referencia a algunos de los más relacionados con esta necesidad.

*ERWin:*

ERWin es una herramienta de diseño de base de datos. Brinda productividad en diseño, generación, y mantenimiento de aplicaciones. Desde un modelo lógico de los requerimientos de información, hasta el modelo físico perfeccionado para las características específicas de la base de datos diseñada, ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de base de datos.

ERWin hace fácil el diseño de una base de datos. Los diseñadores de bases de datos sólo apuntan y pulsan un botón para crear un gráfico del modelo E-R (Entidad-Relación) de todos sus requerimientos de datos y capturar las reglas de negocio en un modelo lógico, mostrando todas las entidades, atributos, relaciones, y llaves importantes.(Anónimo 1996)

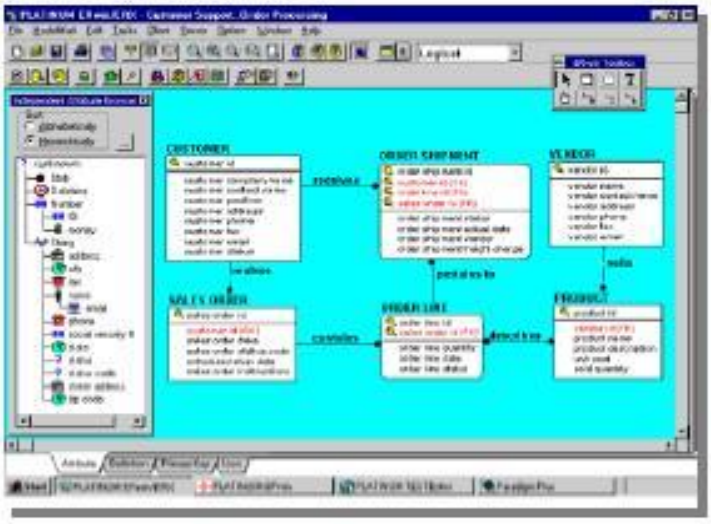

Figura 6. Ventana del ERwin

*EasyCASE:*

EasyCASE Profesional -el centro de productos para procesos, modelado de datos y eventos, e Ingeniería de Base de Datos- es un producto para la generación de esquemas de base de datos e ingeniería inversa que trabaja para proveer una solución comprensible para el diseño, consistencia y documentación del sistema en conjunto. Esta herramienta permite automatizar las fases de análisis y diseño dentro del desarrollo de una aplicación.

EasyCASE permite capturar los detalles de diseño de un sistema y comunicar las ideas gráficamente, para que sean fáciles de ver y entender. Para un diseño legítimo y modelado de

datos, procesos y eventos, permite crear y mantener diagramas de flujo de datos, diagramas de entidad-relación, mapas de estructura y más.(Anónimo 1996)

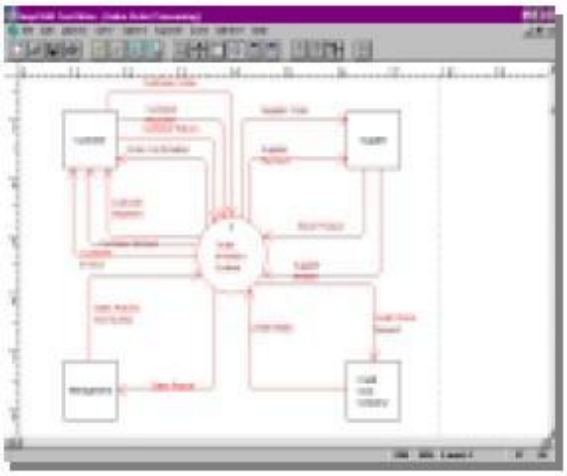

Figura 7. Ventana de EasyCASE

*ERECASE:*

Este es un ejemplo de una herramienta CASE desarrollada en la UCLV, en específico por varios profesores y estudiantes del seminario de Bases de Datos del CEI, liderados por el Dr. Carlos García González.

Como herramienta se propuso implementar no solo lo más característicos de otras herramientas (permitir el modelado entidad-relación de esquemas de bases de datos relacionales), sino además amplió varias posibilidades como soportar el uso de la agregación en diagramas EER; incluir una nueva construcción: la interrelación de asociación con dependencia de existencia; implementar un algoritmo que realiza la validación estructural del esquema conceptual EER; e implementar un nuevo algoritmo que realiza la detección y corrección de inconsistencias de referencias cíclicas. [GARCIA 2010]

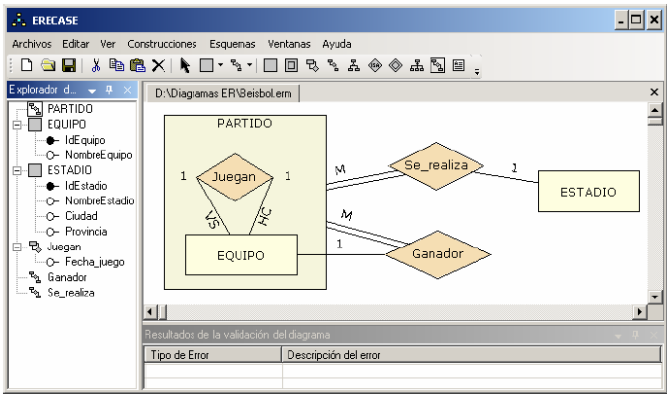


Figura 8. Ventana de ERECASE

También es una herramienta de fácil uso, pero muy dirigida al modelo entidad-relación extendida, el más utilizado en la creación de sistemas de información, que serían fuentes para los almacenes de datos.

Sin lugar a dudas las herramientas CASE han venido a revolucionar la forma de automatizar los aspectos clave en el desarrollo de los sistemas de información, debido a la gran plataforma de seguridad que ofrecen a los sistemas que las usan y es que éstas, brindan toda una gama de componentes que incluyen todas o la mayoría de los requisitos necesarios para el desarrollo de los sistemas, han sido creadas con una gran exactitud en torno a las necesidades de los desarrolladores de sistemas para la automatización de procesos incluyendo el análisis, diseño e implantación.

Estas herramientas quizá puedan ser útiles para modelar diagramas estrella o copo de nieve, etc., pero lo que ninguna incluye es la captación automática de las definiciones relacionales (tablas, atributos e interrelaciones) de los sistemas de información tradicionales, para permitir su conversión a modelo dimensional con el objetivo de crear almacenes de datos. Esta es una de las consideraciones que hace necesaria esta investigación, de la cual este trabajo es una parte inicial.

## 1.8. Lenguaje de Modelado Unificado (UML).

El UML que surge a partir del Método de **Booch**, el Método **OOSE** (*Object-Oriented Software Engineering*, Ingeniería del Software Orientada a Objetos) de *Jacobson* y el Método **OMT** (*Object Modeling Technique*, Técnica de Modelado de Objetos) de *Rumbaugh*, es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. En particular, UML cubre la especificación de todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar un sistema con gran cantidad de software.

UML es actualmente la herramienta más utilizada para el desarrollo del software en todo el mundo y está constituido en su primera versión por 9 diagramas (4 estructurales y 5 de comportamiento), mientras que en su versión 2 incluye 13 diagramas.

A los efectos del desarrollo de este trabajo, que consiste en crear una herramienta CASE o software de ayuda al análisis de sistemas relacionales y su adecuación a almacenes de datos, como quiera que era imposible desarrollar todo el software de una vez y acá se propone una primera versión, se estimo útil desarrollar el modelado del mismo con el uso del UML y en especial los diagramas de casos de uso y actores, clases, actividades y componentes o artefactos. Esto garantiza al menos a nivel del análisis y diseño un proceso evolutivo que además gracias a la

herramienta a utilizar basado en el UML permitirá dar continuidad al software en versiones futuras, ampliando sus posibilidades.

*Diagrama de Clases:* Un diagrama de clases presenta un conjunto de clases, interfaces y colaboraciones, y las relaciones entre ellas, son los diagramas más comunes en el modelado de sistemas orientados a objetos y se utilizan para describir la vista de diseño estática de un sistema.

*Diagrama de Casos de Uso*: Los diagramas de casos de uso se emplean para modelar el contexto del sistema, subsistema o clase, o el modelado de los requisitos de comportamiento de esos elementos. Cada uno muestra un conjunto de casos de uso, actores y sus relaciones.

*Diagrama de Actividades*: Un diagrama de actividades es fundamentalmente un diagrama de flujo que muestra el flujo de control entre actividades, aunque a diferencia de estos puede mostrar concurrencias y ramas de control. Los diagramas de actividades se utilizan para modelar los aspectos dinámicos de un sistema, lo que generalmente implica modelar los pasos secuenciales y concurrentes de un proceso computacional, y también para modelar el flujo de un objeto conforme pasa de un estado a otro en diferentes puntos del flujo de control.

*Diagrama de artefactos*: Un diagrama de artefactos muestra la organización y las dependencias entre un conjunto de artefactos. Los diagramas de artefactos se utilizan para modelar la vista de implementación estática de un sistema. Esto implica modelar las cosas físicas que residen en un nodo, tales como ejecutables, bibliotecas, tablas, archivos y documentos. En la versión 1 de UML y por ende en la herramienta *Rational Rose 2003*, no existe distinción entre artefactos y componentes, por lo que se conocerá como Diagrama de Componentes el que se desarrolla en este trabajo.

## 1.9. Herramientas utilizadas.

### *Rational Rose Enterprise Edition:*

El *Rational Rose* (*Rational Object Oriented Software Engineering*) es una herramienta CASE, que permite el diseño detallado del software y la generación de código fuente (de programas y bases de datos) así como ingeniería inversa (obtención del diseño a partir del código fuente), basado en modelos con soporte UML. Es una forma de ayuda para la comprensión del sistema y de sus distintos componentes. Su característica más significativa consiste en la creación de componentes, que contengan una serie de archivos dentro de los cuales se encuentran las distintas clases pertenecientes a dicho componente. [Rational, 2003].

Este paquete de análisis y diseño, en su versión 2003, será utilizado como base para el desarrollo de los diagramas solicitados y necesarios para el modelado de la herramienta CASE objeto de estudio. El mismo se encuentra instalado adecuadamente en la intranet de MFC.

## *Delphi*

*Delphi* es una potente herramienta de desarrollo de programas que permite la creación de aplicaciones para Windows 3.x, Windows95 y Windows NT, fue desarrollado sobre una plataforma Windows NT Workstation.(Anónimo 2002)

Las aplicaciones pueden colocarse de forma muy sencilla en la pantalla según el principio de módulos. Para ello se dispone de una paleta dotada de una gran variedad de componentes, algo así como los bloques de construcción de cada programa. Esta paleta es denominada por Borland VCL (*Visual Component Library*), o biblioteca de componentes visuales. Tiene un aspecto similar a *Visual Basic*, pero aunque el aspecto externo indica la misma facilidad de uso que este, el corazón del sistema *Delphi* es mucho más potente. *Delphi* dispone del *Object Pascal*, un lenguaje de programación muy poderoso que está sin dudas a la altura del *C++*. Este lenguaje surge a partir del desarrollo del *Borland Pascal 7.0*, un lenguaje que ocupa un lugar muy importante en la programación de ordenadores personales.

Una de las principales características y ventajas de *Delphi* es su capacidad para desarrollar aplicaciones con conectividad a bases de datos de diferentes fabricantes. El programador de *Delphi* cuenta con una gran cantidad de componentes para realizar la conexión, manipulación, presentación y captura de los datos, algunos de ellos liberados bajo licencias de código abierto o gratuito. Estos componentes de acceso a datos pueden enlazarse a una gran variedad de controles visuales, aprovechando las características del lenguaje orientado a objetos, gracias al polimorfismo. En la paleta de componentes pueden encontrarse varias pestañas para realizar una conexión a bases de datos usando diferentes capas o motores de conexión.(Anónimo 2002)

El componente de Delphi utilizado en este trabajo fue el *TXMLDocument* que se instala por defecto en su Paleta de componentes.

## *ADO*

En nuestro caso usamos ADO, ***ActiveX Data Objects*** (**ADO**) es uno de los mecanismos que usan los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas. Con ADO, un programa puede leer, insertar, editar, o borrar, la información contenida en diferentes áreas de almacenamiento dentro de la base de datos llamadas tablas. Además, se puede manipular la propia base de datos para crear nuevas áreas para el almacenamiento de información (tablas), como también alterar o eliminar las ya existentes, entre otras cosas. Fue desarrollado por Microsoft y es usado en ambientes Windows por lenguajes de programación como *Visual Basic*, *C++*, *Delphi* entre otros, como también en la Web mediante el uso de *Active Server Pages* (ASP) y el lenguaje *VBScript*. Los principales componentes de ADO son:

- **Connection** (Permite establecer una conexión con la base de datos)

- **Recordset** (Maneja un conjunto de registros de la base de datos)
- **Command** (Permite enviar órdenes SQL para ser ejecutados por la base de datos).(Anónimo 2005)

La **conexión** es como una autopista que permite el flujo de datos entre el programa y la base de datos. Por ella pueden viajar las órdenes que desde el programa se usan para hacer solicitudes de información a la base de datos o para realizar una operación dentro de ella como borrar registros, añadir registros, modificar tablas, etc. También, por esta autopista, pueden ir y venir los datos, desde y hacia la base de datos, entre otras cosas. Tanto el *recordset* como la orden usan la conexión para comunicarse con la base de datos. La conexión se comunica con la base de datos a través de un intermediario llamado "proveedor de datos".

El *Recordset* es, como su nombre lo indica, un conjunto de registros. En general, sus datos tienen su origen en una base de datos, aunque también pueden generarse independientemente de ésta. Un *recordset* puede contener cero o más registros. Cada *recordset* tiene una colección de campos, que es común a todos los registros. Podemos verlo como una matriz o tabla, en donde las filas son los registros, y las columnas son los campos.

En el programa desarrollado en este trabajo se utilizó como herramienta para la conexión a la base de datos y se utilizaron sus funciones (*OpenSchema* es el nombre de la función) para lograr la estructura de la base de datos y las características de sus campos. [6]. Para la salva de los datos utilizamos el formato XML que es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados.(Anónimo 2007)

## 1.10. Conclusiones parciales.

En este capítulo se ha tratado todo lo relacionado al estado del arte sobre el tema objeto de estudio.

Se abordaron los conceptos y características principales de los almacenes de datos y del modelo dimensional que es más afín a estos, incluyendo los métodos más utilizados en la construcción de los mismos, sus cualidades y las variantes de modelado de estos.

A su vez se detallaron algunas consideraciones sobre los modelos relacionales y los gestores de datos, muy utilizados en el desarrollo y explotación de los sistemas de información tradicionales, que son la fuente de los datos que poblaran a los almacenes. Por otra parte se hizo un breve esbozo de algunos CASE usados en el modelado de bases de datos, de los cuales se consideraron algunas características típicas que debía cumplir el software que se presenta.

Como conclusión de todo esto deben especificarse las pautas de conversión que se consideren importantes para esta primera versión de la herramienta que se propone.

También hace referencia al lenguaje de modelado del software utilizado (UML), especificando que diagramas se exponen en el capítulo correspondiente; y las herramientas que se usaron en este trabajo para la implementación del CASE propuesto.

# Capítulo 2.

# "Análisis y Modelado de la Herramienta CASE DWH-Inspector"

# Capítulo 2. “Análisis y Modelado de la Herramienta CASE DWH-Inspector”

En este capítulo se describen diversos requerimientos que debe cumplir la aplicación. También se planteará el análisis del sistema utilizando el Lenguaje Unificado de Modelado (UML), que permitirá representar diferentes diagramas.

## 2.1. Especificación de los requisitos de software.

### Requerimientos funcionales

Un Requerimiento Funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Los requerimientos funcionales que debe cumplir este sistema se describen como pautas a continuación, a partir de las cuales se debe establecer el algoritmo programático correspondiente:

1. El software debe ser capaz de captar los metadatos de una base de datos (tablas, atributos, tipos de atributos, etc.) de diferentes SGBDR. En específico para esta primera versión se logra la implementación de la lectura de bases de datos .MDB (de *Microsoft Access*) y de cualquier otro tipo de base de datos cuyos *drivers* estén instalados en los Orígenes de Datos (ODBC).
2. En esta primera versión se toman en cuenta los campos de tipo numérico (utilizados para almacenar datos mensurables, aditivos o semiaditivos) y de tipo fecha (por una consideración especial de introducir la dimensión TIEMPO en el almacén que se proponga, para considerar posibles tablas de hecho; mientras que se toman en cuenta los campos de tipo caracteres o texto (pero no de texto amplio como los de tipo memo), que pueden ser descripciones dimensionales de los hechos.
3. No se toman en cuenta otros tipos de datos que en una versión más avanzada deben tomarse en cuenta, como los campos lógicos, de hipervínculo, etc., que quizá puedan ser usados como hechos de tipo contadores.
4. Tampoco fue posible identificar claramente dentro de los campos textuales cuales de ellos pudieran agruparse en dimensiones de LOCALIZACIÓN (o geográficos) u otras que son muy características en los almacenes de datos.
5. Si una tabla estudiada incluye campos numéricos que puedan ser aditivos o semiaditivos, así como campos fecha, se determina que es probable que pueda convertirse en una tabla de hechos y como tal se marca de cierta forma.
6. Si la tabla estudiada no incluye campos numéricos o de tipo fecha, pero incluye claves o códigos que aparecen en las tablas propuestas como tablas de hechos, y además incluye

campos de caracteres, entonces puede ser considerada como tabla dimensional y como tal se marca de cierta forma diferente a las anteriores.

7. Si la tabla estudiada no cumple con las condiciones de los pasos 2 y 3, entonces no se toma en consideración para la propuesta del modelo dimensional y no se marca en ninguna de las variantes.
8. El software debe ser capaz de presentar de manera visual el resultado de este algoritmo de marcas, así como destacar de cierta forma los diferentes tipos de atributos tomados como base del análisis.
9. El software debe ser capaz de permitir que el usuario-analista pueda ajustar la propuesta, cambiando la tipología de alguna de las tablas u ocultándola, así como de algunos atributos dentro de las tablas.
10. El software debe ser capaz de, a partir de una tabla propuesta como de hechos marcada, generar algunos modelos dimensionales conocidos (específicamente el esquema estrella o el de copo de nieve).
11. El software debe ser capaz de guardar de cierta forma el resultado alcanzado por el analista para su posterior uso como almacén de datos.

## Requerimientos no funcionales

Los requerimientos no funcionales especifican propiedades o cualidades que el producto de software debe tener, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma y facilidad de mantenimiento.

En el presente trabajo se considera que el software debe ser de tipo monousuario, a ser explotada en una microcomputadora convencional INTEL compatible, a partir de un ejecutable, sin requerimientos especiales de rendimiento, velocidad, etc. Sin embargo como es una primera versión de un producto que se propone expandir posteriormente, debe ser modelado y programado con estilo de componentes o módulos fácilmente mantenibles, cohesivos y poco acoplados.

## Requerimientos de Seguridad

- La información que se encuentre en las bases de datos relacionales que se lean para estudiar sus características, no será revelada por el software, de manera que algún usuario-analista no pueda tener conocimientos que no le correspondan. Solo se analizarán los metadatos.
- La propuesta de clasificación de las tablas como dimensionales o de hechos, así como las adecuaciones que de forma controlada (manual) el usuario-analista establezca, será almacenada en un archivo XML.
- El uso y manejo del sistema se supone que sea controlado por el usuario-analista que lo explote. No se requieren otros niveles de seguridad específicos.

## Requerimientos de Confiabilidad

- Todas las salidas del sistema tendrán 100% de precisión, acorde con los pasos establecidos o las decisiones del analista-usuario.

## Requerimientos de Rendimiento

- No existen restricciones específicas de este tipo.

## Requerimientos de apariencia o interfaz externa

- La interfaz será agradable, sencilla y sugerente como corresponde a una aplicación de escritorio, asumiendo por defecto el color establecido por el ambiente Windows del usuario-analista.
- Los textos que muestran las acciones a realizar en el menú o en la ventana central de resultados tendrán un tipo de letra y tamaño adecuados.
- Se emplearán íconos sencillos para marcar los tipos de tablas y los atributos a destacar como Tablas de Hechos o Dimensionales, y tipos de campos numérico y de fechas.
- El sistema será diseñado para una resolución de pantalla superior o igual a 800 X 600 píxeles.
- El sistema incluirá una opción optativa para imprimir los metadatos del sistema de información que sea objeto de estudio.

## Requerimientos de Hardware

La PC utilizada debe tener como mínimo las siguientes características:

- Computador PENTIUM III o superior.
- 256 MB de memoria RAM o más (según exija el Sistema Operativo instalado).
- Una Impresora para la impresión de los Informes, si se desea.

## Requerimientos de Software

- Las estaciones de trabajo clientes utilizarán como sistema operativo cualquier versión de Windows superior a Windows XP con MDAC (*Microsoft Data Access Components*) instalado.

## Requerimientos de Soporte

- El sistema contará con una documentación apropiada para agilizar su explotación, mantenimiento y configuración.

## Requerimientos de Facilidad de Uso

- La interfaz será fácil de usar para los diversos usuarios que interactúen con ella.

- El sistema estará bien documentado, con el fin de lograr un mejor uso de los servicios que este ofrecerá, para ello se realizará una ayuda que explique paso a paso cada una de las funcionalidades del software.

Requerimientos legales

- Los módulos de desarrollo del proyecto *DWH-Inspector* así como la documentación del mismo, son propiedad de la UCLV y específicamente de la facultad de MFC.

## 2.2. Casos de Uso.

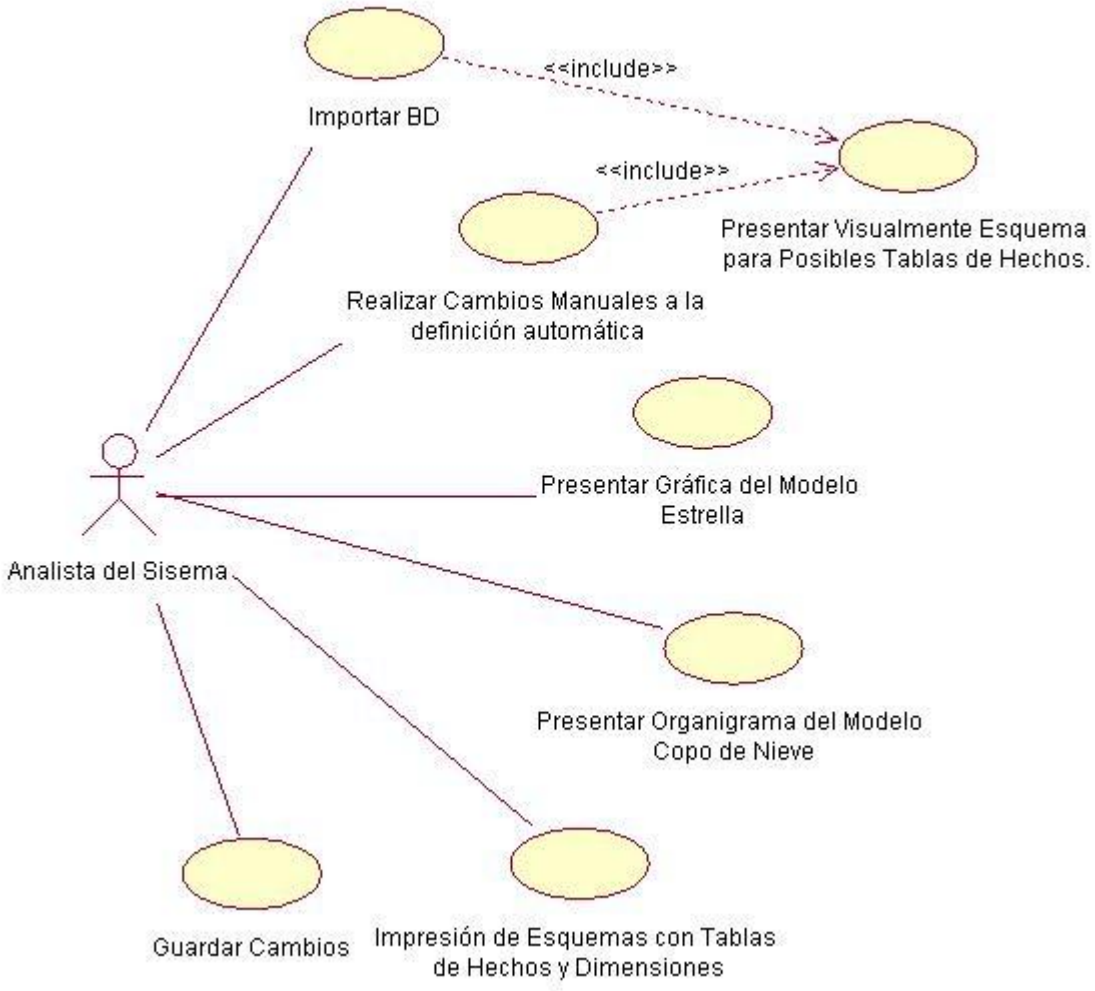


Figura 9. Diagrama de Casos de Uso.

1. Importar Base de Datos.
2. Presentar Esquema para posibles Tablas de Hechos y Dimensiones.
3. Realizar cambios manuales a la definición automática.

4. Presentar Grafico del Modelo Estrella.
5. Presentar Organigrama del Modelo Copo de Nieve.
6. Impresión de Esquema con tablas de hechos y dimensiones.
7. Guardar Cambios.

## Descripción de los Casos de Uso

| | |
|---|---|
| **Nombre del CU** | Importar Base de Datos. |
| **Actor(es)** | Analista |
| **Propósito** | Un usuario importa la estructura de una Base de Datos de un Sistema de Información típico para hacer un análisis exhaustivo de los metadatos de la misma. |
| **Descripción:** | El caso de uso de inicia cuando el actor hace clic sobre el menú archivo y su submenú importar Base de Datos, seguidamente se muestra una ventana para seleccionar el proveedor y origen de los datos y a continuación se muestran los datos leídos después de procesados en la ventana principal del programa |
| **Referencia** | Es necesario tener instalado el proveedor correspondiente para cada Base de Datos |
| **Precondiciones** | - Tener el Data Access Component instalado(MDAC)<br>- Que la Base de Datos sea válida. |

| | |
|---|---|
| **Nombre del CU** | Presentar Esquema para posibles Tablas de Hechos y Dimensiones. |
| **Actor(es)** | Analista |
| **Propósito** | Obtener propuestas de las tablas de hechos y de dimensiones que tiene la Base de Datos. |
| **Descripción:** | El caso de uso se inicia cuando el actor obtiene las tablas de hechos y dimensiones, las cuales son mostradas visualmente después de importar una Base de Datos o de abrir un proyecto salvado con anterioridad. |
| **Precondiciones** | - Haber importado una Base de Datos<br>- Tener un proyecto abierto. |

| | |
|---|---|
| **Nombre del CU** | Realizar cambios manuales a la definición automática. |
| **Actor(es)** | Analista |
| **Propósito** | El usuario decide si el sistema hizo una elección correcta o no. |

| | |
|---|---|
| **Descripción:** | El caso de uso se inicia cuando el actor revisa las tablas de hechos y dimensiones que el sistema propone, seguidamente decide si esta propuesta del sistema es correcta, en caso de no ser así procede a cambiarlas de acuerdo a su criterio. Puede además ocultar los atributos y/o tablas que no sean de su interés. |
| **Referencia** | CU_Presentar Esquema para posibles Tablas de Hechos y Dimensiones. (*Extend*) |
| **Precondiciones** | - Haber definido tablas de Hechos y Dimensiones. |
| **Postcondiciones** | - Salvar el proyecto si sus datos fueron modificados. |

| | |
|---|---|
| **Nombre del CU** | Presentar Gráfico del modelo estrella. |
| **Actor(es)** | Analista |
| **Propósito** | Obtener el diagrama estrella de una base de datos seleccionada. |
| **Descripción:** | El caso de uso se inicia cuando el actor después de haber importado la Base de Datos y después de haber hecho los cambios pertinentes en la estructura originalmente propuesta de dicha Base de Datos, procede a presentar el gráfico del modelo estrella de una de las tablas marcadas como posibles de hechos, propuesta para dicha Base de Datos. |
| **Precondiciones** | Haber hecho los cambios manuales que el usuario considere a la definición automática. |

| | |
|---|---|
| **Nombre del CU** | Presentar Organigrama del modelo Copo de Nieve. |
| **Actor(es)** | Analista |
| **Propósito** | Obtener el Organigrama de Copo de Nieve |
| **Descripción:** | El caso de uso se inicia cuando el actor después de haber importado la Base de Datos y después de hacer hecho los cambios pertinentes en la estructura original de dicha Base de Datos, procede a presentar el Organigrama de Copo de Nieve correspondiente a una de las tablas marcada como de hechos en el modelo propuesto para dicha Base de Datos. |
| **Precondiciones** | Haber hecho los cambios manuales que el usuario considere a la definición automática. |

| | |
|---|---|
| **Nombre del CU** | Impresión de Esquema con tablas de hechos y dimensiones |
| **Actor(es)** | Analista |
| **Propósito** | Imprimir las tablas de Hechos y Dimensiones que presenta el sistema. |

| | |
|---|---|
| **Descripción:** | El caso de uso se inicia cuando el actor después de importar la Base de Datos y de mostrar las tablas de Hechos y Dimensiones que contiene dicha Base de Datos procede a la impresión del esquema de las tablas antes mencionadas. |
| **Precondiciones** | Haber importado la Base de Datos y presentado la propuesta de sus correspondientes tablas de Hechos y Dimensiones. |

| | |
|---|---|
| **Nombre del CU** | Guardar cambios. |
| **Actor(es)** | Analista |
| **Propósito** | Guardar los cambios que se hayan realizado en el proyecto. |
| **Descripción:** | El caso de uso se inicia cuando el actor después de haber seleccionado y ocultado las tablas y campos que no hayan sido de su interés, procede a salvar el proyecto para no perder dichos cambios. Además puede salvar el proyecto luego de la importación del mismo sin haber hecho modificación alguna. |

## 2.3. Clases

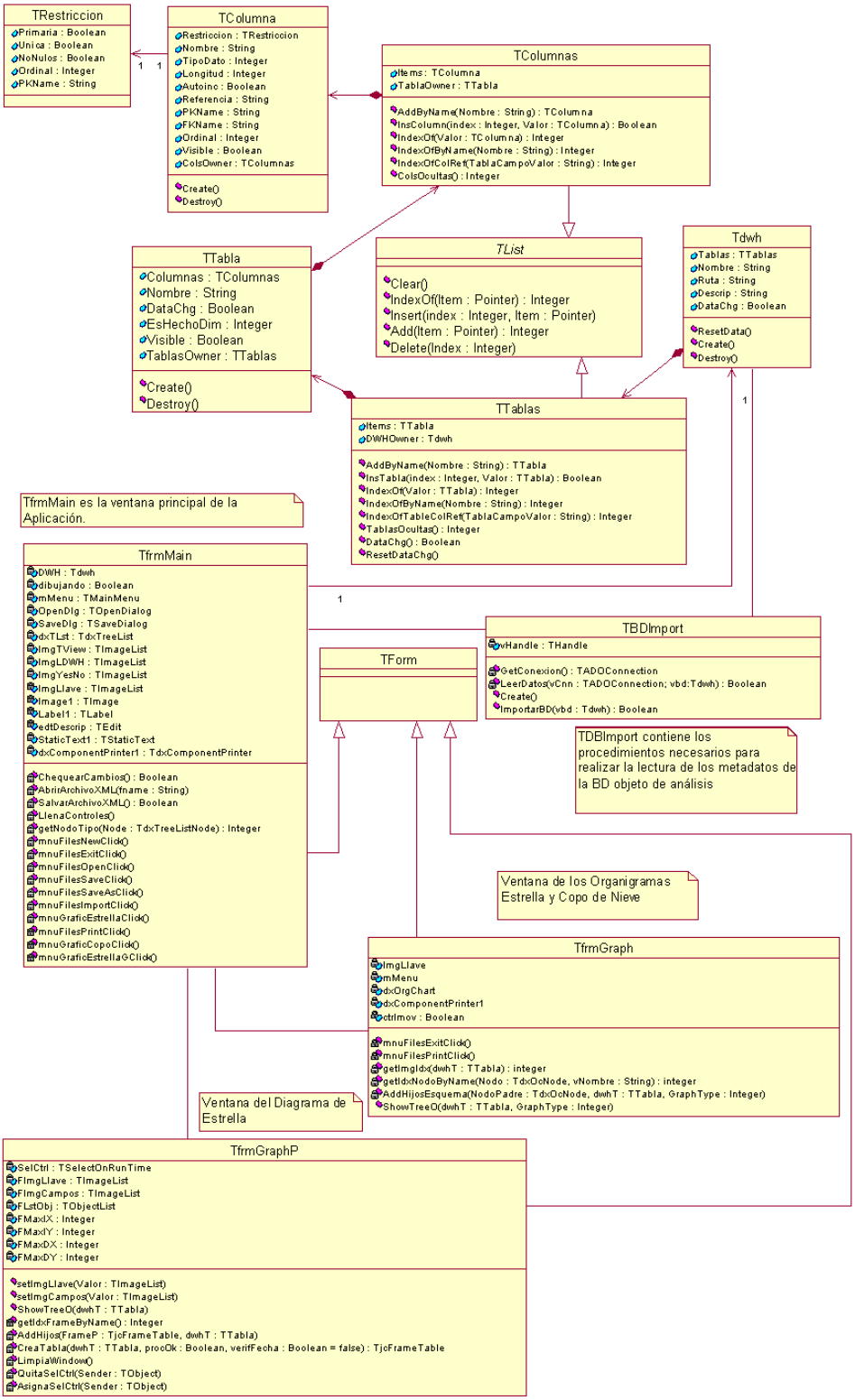


Figura 10. Diagrama de Clases.

## Descripción de las Clases

A continuación se realizará una descripción de las clases más importantes del Sistema.

**TRestriccion**: Esta Clase se creó para albergar en sus variables los atributos como: si es llave o no, si el campo debe contener un valor único, si admite Nulos, etc.

**TColumna**: Esta Clase contiene una variable de tipo **TRestriccion** y además datos sobre la columna como: *Nombre*, *Tipo de dato, Longitud, Nombre de llaves primarias y/o foráneas* en caso de contenerlas, se adiciona a esos datos una variable booleana llamada **visible** que se utilizará para saber cuando se mostrará visualmente el campo o no. Los métodos **Create() y Destroy()** (heredados de **TObject**) se implementan para crear y destruir la variable **Restriccion** de tipo **TRestriccion**.

**TColumnas**: Esta clase hereda de la clase abstracta **TList** y está diseñada para albergar una lista de las *Columnas* de una tabla. Se agregaron funciones como: **AddByName(), InsColumns() e IndexOfByName()** para facilitar el trabajo con las columnas haciendo referencia a su nombre. **TablaOwner** es un puntero a la instancia de la clase Tabla que contiene la lista de columnas, **ColsOcultas()** devuelve un entero equivalente al número de columnas que el usuario ocultó.

**TTabla**: Contiene los datos de las tablas incluyendo una lista de Columnas del tipo **TColumnas**. La variable **visible** se utilizará para saber cuando se mostrará visualmente la tabla y **EsHechoDim** contendrá un valor entero que define si la tabla es de Hecho, Dimensiones o ninguna de las dos. La variable **TablasOwner** es un puntero a la instancia de la clase **TTablas** que la contiene.

**TTablas**: Esta clase hereda de la clase abstracta *TList* y alberga la lista de Tablas de la Base de la Datos. Se agregaron funciones como: **AddByName(), InsTabla() e IndexOfByName()** para facilitar el trabajo con las tablas haciendo referencia a su nombre. **DWHOwner** es un puntero a la instancia de la clase *Tdwh* que contiene la lista de tablas. **TablasOcultas()** devuelve un entero equivalente al total de tablas que ocultó el usuario y **DataChg()** devuelve verdadero si se realizaron cambios en los datos.

**Tdwh**: Esta Clase contiene los datos generales del proyecto. Una variable de tipo **TTablas** que contendrá lo importado de la base de datos, así como **Nombre** y **Ruta** de dónde se encuentra ubicado dicho proyecto. **DataChg** retornará verdadero si se realizó variación alguna a nivel de proyecto.

**TBDImport**: Esta clase es una de las principales del programa, en ella se han encapsulado los procedimientos necesarios para lograr importar los metadatos de la base de datos objeto de estudio. **ImportarBD()** es una función pública desde la cual se invoca el proceso de importar, devuelve verdadero si fue exitoso. La función **GetConexion()** devuelve un objeto de tipo *TADOConnection*, cuya cadena de conexión se crea utilizando la función **PromptDataSource()**

(que se encuentra en la *Unit ADODB*) que luego se le pasa como parámetro junto a una variable de tipo Tdwh a **LeerDatos()** que realiza el proceso de lectura de los metadatos y salva de los mismos en dicha variable. El decidir si la tabla es de Hechos o Dimensiones está a cargo del procedimiento **ChkWH(),** privado a la *Unit uImport*, que es la que contiene la declaración de la clase **TBDImport**.

**TfrmMain** (Ventana Principal): Esta clase hereda de *TForm* y se corresponde con la interfaz de usuario. Contiene componentes de tipo *TOpenDialog* y *TSaveDialog* utilizados respectivamente para abrir y salvar los proyectos, se utilizan en los procedimientos privados **AbrirArchivoXML()** y **SalvarArchivoXML()**, **dxTLst** (del tipo *TdxTreeList*) es el *treeview* que muestra visualmente las propiedades de las tablas y campos. **ImgTView**, **ImgLDWH**, **ImgYesNo** e **ImgLlave**, todos del Tipo *TImageList* contienen listas de Imágenes utilizadas por **dxTLst**. Los eventos **OnClick()** de cada *TMenuItem* que contiene la variable *mMenu* del tipo *TMainMenu* están asociados a procedimientos que comienzan con el prefijo **mnu** y terminan con el sufijo **Click**, ejemplo: **mnuGraficEstrellaGClick() y mnuFilesSaveAsClick().** La función **ChequearCambios()** devuelve verdadero si se realiza una salva exitosa luego de detectar cambios en el proyecto del cual se va a salir. **LlenaControles()** realiza el llenado del *treeview* con los datos contenidos en la variable DWH del tipo **Tdwh**.

**TfrmGraphP** (Diagrama Estrella): Una instancia de esta Clase se crea al puntear el submenú cuya propiedad *Caption* es igual a **"Diagrama Estrella"** y se hace visible luego de ejecutar el método público **ShowTreeO()** al cual se le pasa como parámetro la tabla de hechos objeto de análisis. Dinámicamente se crea un control del tipo **TjcFrameTable** que se sitúa en el centro superior de la pantalla invocando al procedimiento privado **CreaTabla()**, a continuación mediante el método **AddHijos()** se muestran en pantalla las tablas de Dimensiones asociadas a dicha tabla de hechos. Las variables privadas *FMaxIX, FMaxIY, FMaxDX, FMaxDY* de tipo entero se utilizan para manipular las coordenadas donde se situarán las tablas. Cada vez que se crea un control del Tipo **TjcFrameTable** mediante el procedimiento **CreaTabla()** se le asignarán a los eventos **OnCick**() y **OnExit()** los procedimientos **AsignaSelCtrl()** y **QuitaSelCtrl()** respectivamente, que garantizarán asignarle al componente *SelCtrl* del tipo *TSelectOnRunTime* para posibilitar su movimiento.

## 2.4. Actividades.

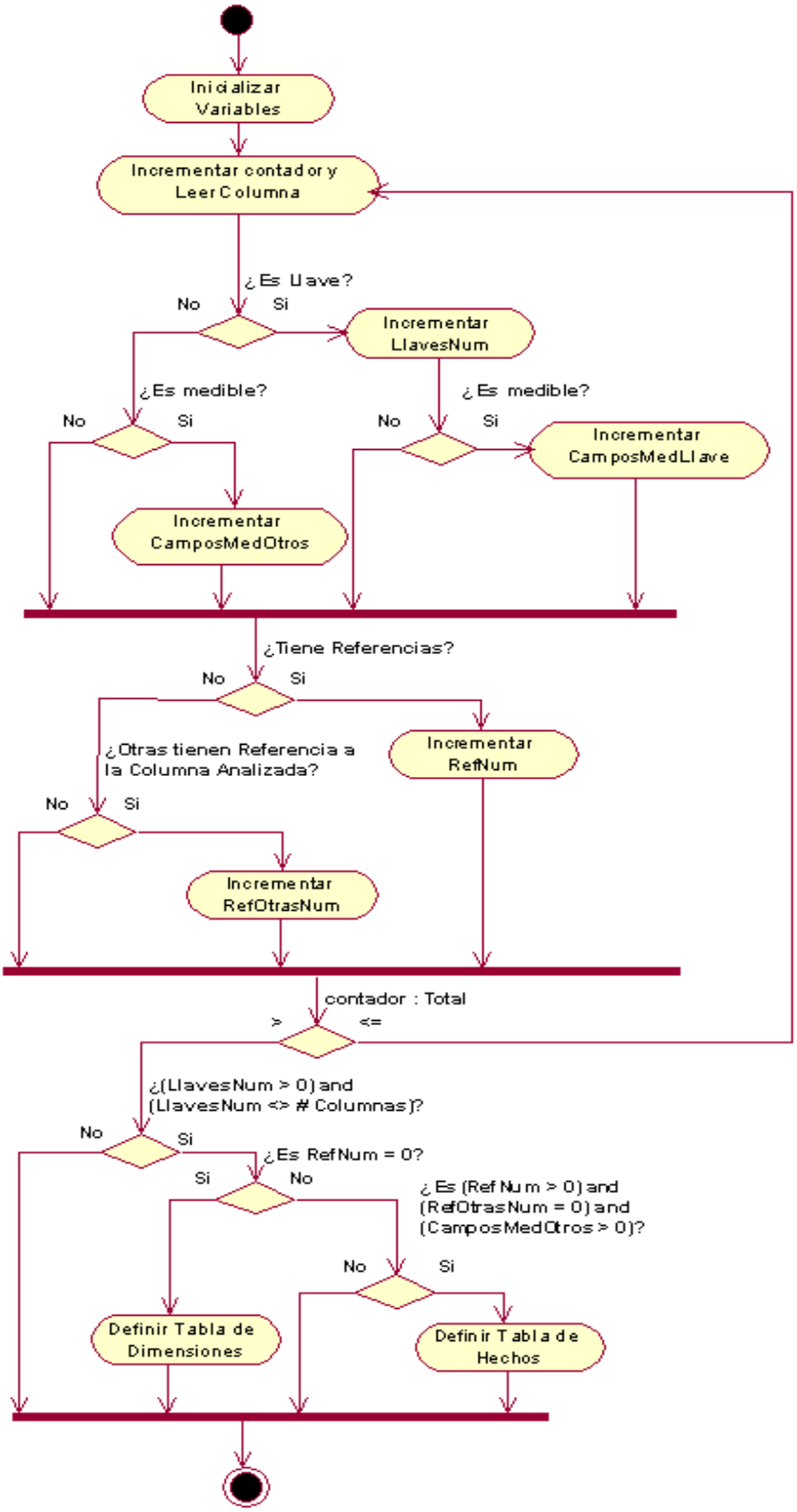


Figura 11. Diagrama de Actividades.

## Descripción de las Actividades de este diagrama

Código utilizado:

```
// Chequear los datos de diseño de una tabla
// Para definir si la misma es de Hechos o dimensiones
procedure ChkWH(vntabla: TTabla);
var
  LlavesNum, RefNum, RefOtrasNum: integer;
  i, CamposMedLlave, CamposMedOtros: integer;
begin
  LlavesNum:= 0;
  RefNum:= 0;
  RefOtrasNum:= 0;
  CamposMedLlave:= 0;
  CamposMedOtros:= 0;

  for i:=0 to vntabla.Columnas.Count -1 do begin
    if vntabla.Columnas[i].Restriccion.Primaria then begin
      inc(LlavesNum);
      if IsMedibleDATA_TYPE(vntabla.Columnas[i].TipoDato) then
        inc(CamposMedLlave);  //Campos Medibles en las llaves
    end else begin
      if IsMedibleDATA_TYPE(vntabla.Columnas[i].TipoDato) then
        inc(CamposMedOtros);  //Campos Medibles que no son llave
    end;
    if (vntabla.Columnas[i].Referencia <> '') then
      //Para ver si existe en esta tabla referencia a otras Tablas
      inc(RefNum)
    else begin
      //Para ver si existe en otras Tablas referencia a este campo
      if vntabla.TablasOwner.IndexOfTableColRef(vntabla.Nombre + '.' +
        vntabla.Columnas[i].Nombre) > -1 then
        inc(RefOtrasNum);
    end;
  end;
```

```
if (LlavesNum > 0) and (LlavesNum <> vntabla.Columnas.Count) then
  if (RefNum = 0) then
    vntabla.EsHechoDim:= 2
  else if (RefNum > 0) and (RefOtrasNum = 0) and
        (CamposMedOtros > 0) then
    vntabla.EsHechoDim:= 1;

end;
```

## 2.5. Componentes.

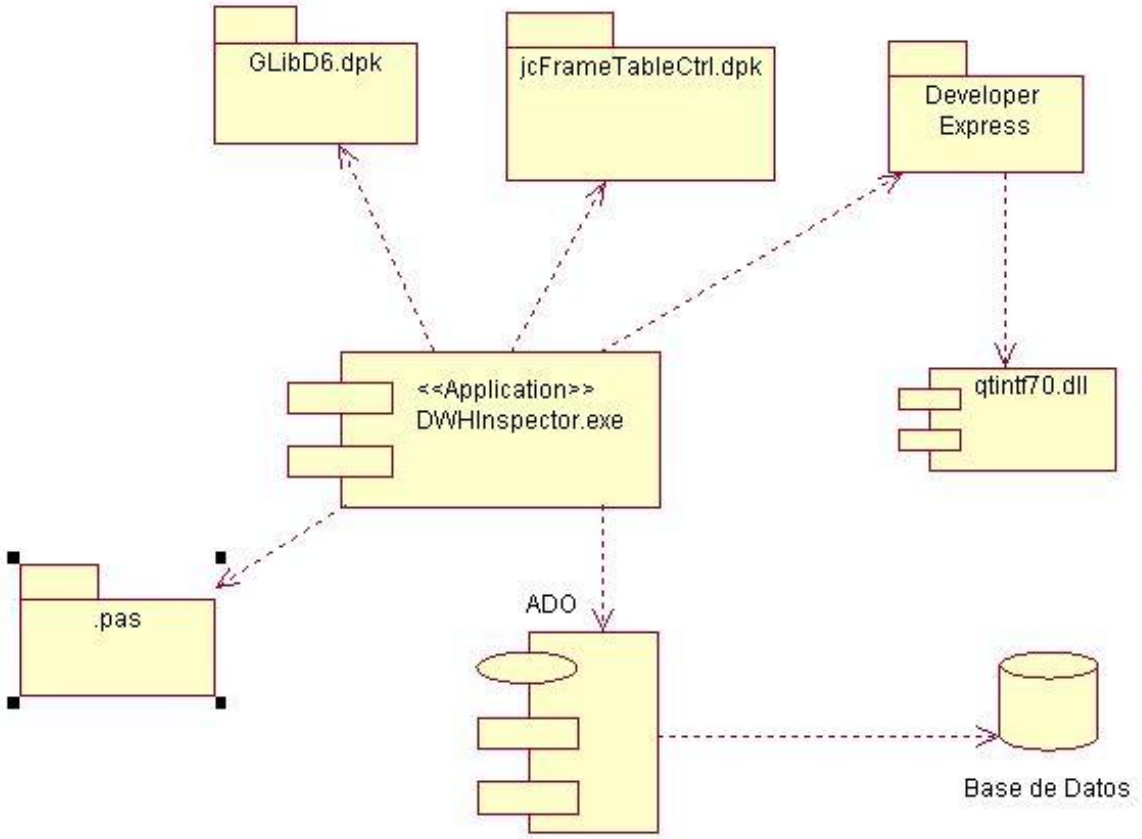


Figura 12. Diagrama de Componentes.

### Descripción de los Componentes

La aplicación consta de un ejecutable que se nombra **DWHInspector** está relacionada con los paquetes GlibD6, jcFrameControl y Developer Express utilizados para las diferentes acciones que realiza la aplicación, el paquete Developer Express se relaciona con la .dll debido a que su funcionamiento depende mucho de este componente, la fuente de dicha aplicación es el paquete

.pas con el cual se relaciona, además la aplicación se relaciona con el componente ADO que usamos para conectarnos a la Base de Datos seleccionada.

## 2.6. Conclusiones parciales.

En este capítulo se trató todo lo relacionado al modelado y análisis del sistema, comenzando por especificar una serie de requerimientos del software encomendado a desarrollar por el tutor.

Se identificó el actor y los casos de uso del software, plasmándolos en el diagrama de casos de uso, y describiéndolos posteriormente.

Posteriormente se modelan y diseñan las clases principales del software, las cuales fueron presentadas en un Diagrama de Clases parcial y descritas a continuación.

De una de las clases se tomo una operación interesante, cuyo algoritmo se presentó en un Diagrama de Actividades.

Finalmente se presenta un Diagrama de Componentes (fuentes, etc.) que conforman la aplicación que finalmente compilada darán lugar al programa ejecutable correspondiente.

# Capítulo 3.

# “Descripción de Características del CASE”.

# Capitulo 3. “Descripción de Características del CASE”.

## 3.1. Aspectos de la Implementación y la Implantación.

El sistema diseñado tiene como nombre DWH-Inspector, dado que inspecciona los metadatos de un sistema de información en modelo relacional para a partir de las pautas explicadas sugerir posibles tablas de hechos y dimensionales para la creación de la estructura dimensional de un Almacén de Datos.

El software desarrollado como resultado de este trabajo fue programado en lenguaje Borland Delphi 7, utilizando las bondades del *Data Access Component* (MDAC) para mediante una conexión por ADO obtener la estructura de una base de datos previamente diseñada. Esta conexión a la Base de Datos será posible siempre que el controlador (*driver*) para dicha conexión este instalado en el Sistema Operativo de la microcomputadora del usuario.

Se utilizaron las bibliotecas de componentes *Developer Express Inc*, así como componentes desarrollados por los autores de este trabajo para poder representar los diagramas dimensionales.

El programa luego de haber importado la estructura de una Base de Datos propone según criterios lógicos qué tablas se proponen como de hechos o de dimensiones y permite al usuario cambiar dicha definición, en caso de que la propuesta del programa no esté acorde a lo esperado.

A partir de una de las tablas propuestas como hechos y que el usuario marque, muestra el diagrama de estrella correspondiente y además organigramas en estrella y copo de nieve que ayudan a conocer con más detalle las relaciones entre las tablas (la de hecho y las dimensionales asociadas).

Por otra parte permite guardar la definición alcanzada y también sacar copia por impresora de los datos que brinda el sistema en forma de gráfico.

Una breve explicación de parte del código de programación utilizado fue presentado al explicar el Diagrama de Actividades en el capítulo anterior.

## 3.2. Explotación de la herramienta CASE DWH-Inspector.

### Ventana Principal

El software que se presenta es una herramienta CASE monousuaria, que al ser invocada presenta la siguiente ventana de inicio:

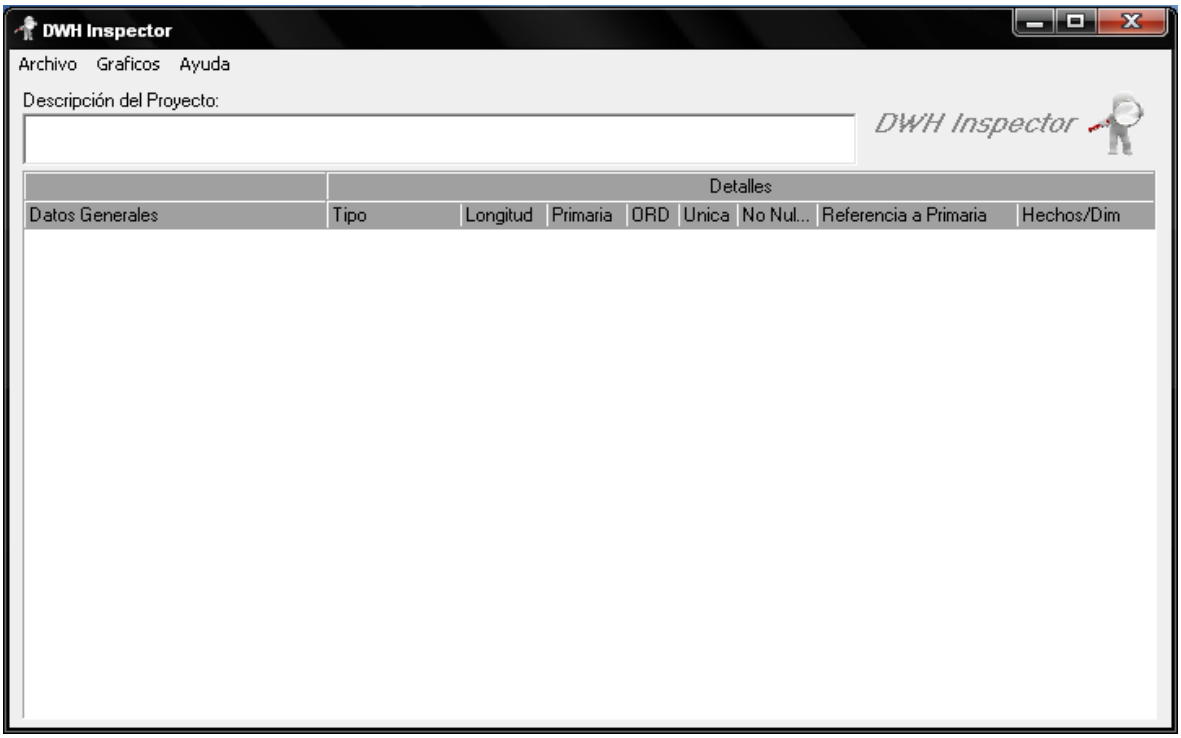


Figura 13. Pantalla Inicial del DWH-Inspector.

La ventana principal muestra una vista inicial de este sistema y los menús disponibles para que el usuario pueda interactuar fácilmente con la aplicación. El menú principal incluye las opciones de **Archivos**, **Gráficos** y **Ayuda**. Si se puntea la opción Archivos, se presenta el siguiente menú de ventana:

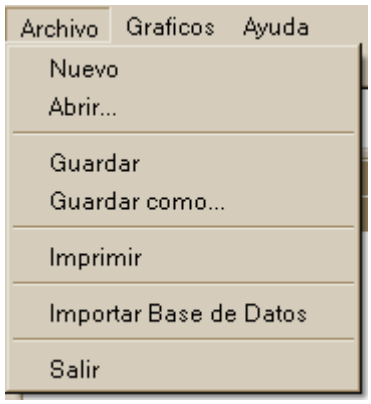


Figura 14. Submenú Archivo.

Acá las opciones **Nuevo** y **Abrir** se corresponden con proyectos de propuesta de Almacenes de datos. Es bueno señalar que **Abrir** buscaría en el directorio correspondiente archivos .DWHI, que son los que se crean cuando se usa la opción **Guardar**. Acá también aparecen las opciones de **Importar Base de Datos**, que es la principal, pues a partir de ahí se inicia la labor de

reconocimiento de una bases de datos relacional existente, e **Imprimir** de la que se ha explicado anteriormente en esta tesis. Por último está la opción de **Salir** para abandonar el programa.


Figura 15. Submenú Gráficos.

Este submenú permite obtener los resultados principales: los diagramas dimensionales propuestos, de manera gráfica. El otro submenú Ayuda, solo presenta por ahora una ventana que indica que es una primera versión de uso docente de la UCLV.

En la Pantalla Inicial aparece un *TreeView* con **Datos Generales**, **Tipo**, **Longitud**, **Primaria**, **ORD**, **Única**, **No Nul..**, **Referencia a Primaria**, **Hechos/Dim**. Debajo de los Datos Generales aparecerán las tablas de la Base de Datos seleccionada con sus atributos correspondientes, el Tipo significa el tipo de atributo que es, la longitud es la longitud de dicho atributo, Primaria significa si el atributo es llave o no, ORD es el número ordinal que tiene el atributo en la Base de Datos, Única significa si el atributo es único o no, No Nul significa si el atributo es nulo o no, Referencia a Primaria quiere decir si dicho atributo hace referencia a alguna llave de alguna otra tabla, y debajo de Hechos/Dim se exponen unas marcas que señalan la propuesta del software en cuanto a que tablas de la Base de Datos original deben ser consideradas tablas de hechos ( )y cuales deben ser consideradas dimensiones ( ).

## Importar Base de Datos

Al puntear la opción **Importar Base de Datos** el sistema muestra la ventana de los orígenes de datos instalados en la PC, aquí el usuario escogerá el proveedor que esté acorde al tipo de motor de datos necesario para lograr una conexión satisfactoria. Por defecto el programa muestra el Microsoft Jet 4.0 OLE DB Provider, que permite obtener la estructura de un sistema de información desarrollado sobre el SGBD Microsoft Access.


Figura 16. Ventana de los orígenes de datos instalados en la PC.

El segundo paso será escoger la ruta de los datos (en caso de que sea un archivo), el servidor o el DSN (*Data Source Name*) creado en ODBC. O sea, se puntea la ficha Conexión y allí se selecciona una base de datos, como se muestra en la figura que sigue.

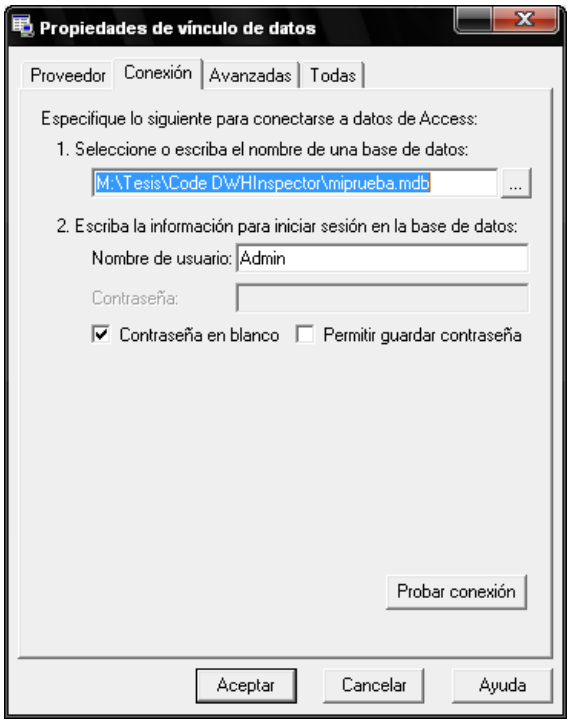


Figura 17. Ficha Conexión para especificar la Base de Datos origen.

Si la conexión es satisfactoria con la cadena devuelta en el paso anterior seguidamente se realizarán consultas a las tablas del sistema utilizando la función *OpenSchema* que permitirán conocer la estructura interna de la Base de Datos objeto de estudio.

## Mostrar Estructura Importada

A partir de la lectura del esquema de la base de datos indicada, y aplicando las pautas de clasificación anteriormente presentadas, el programa presenta la estructura de la base relacional original, marcando indistintamente las tablas como de hechos o dimensionales y también presentando si se desea los atributos de algunas tablas con los datos ya mencionados y con cierto destaque, si son de tipo numérico (azul) o de fechas (verde). También se destacan en color rojo los campos claves. Esto se puede apreciar en la siguiente figura.

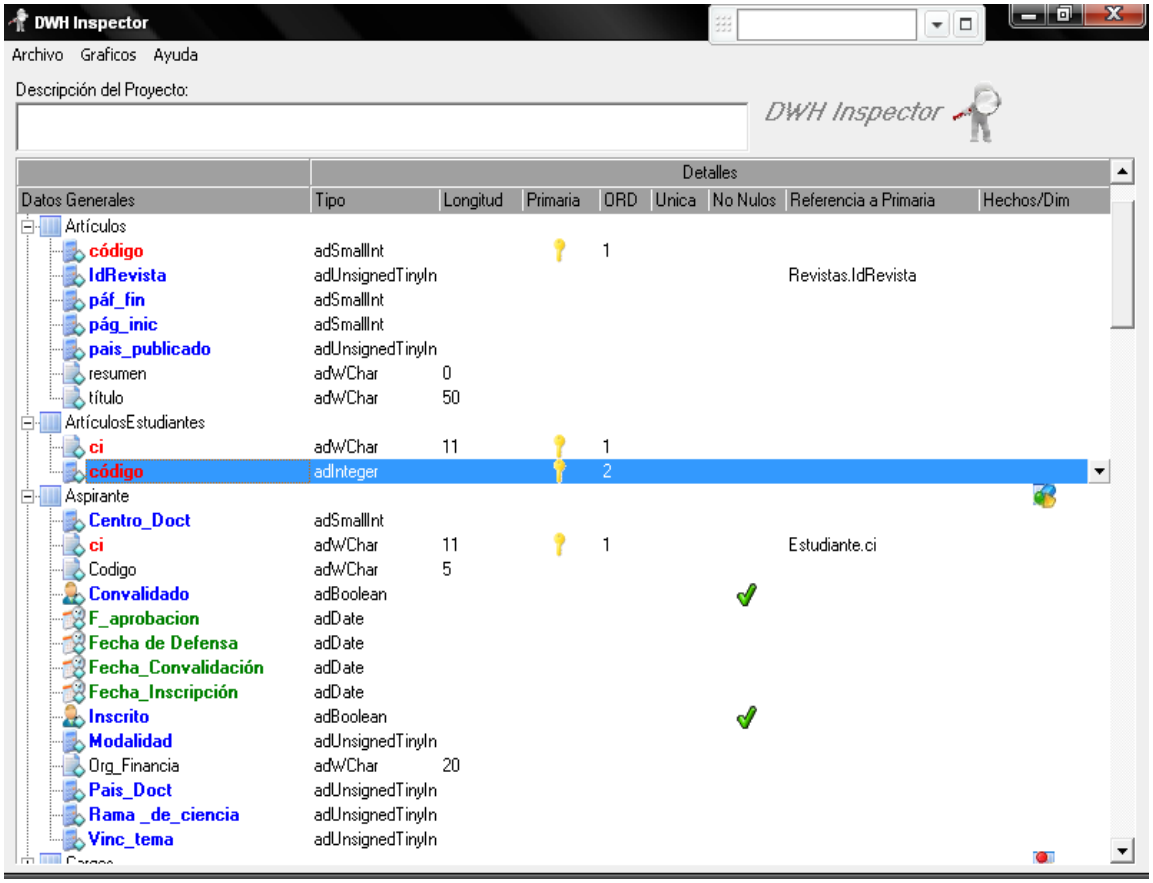


Figura 18. Ventana Principal con la presentación de la estructura de una Base de Datos.

Como se puede apreciar algunas tablas no quedan marcadas como ninguno de los tipos fundamentales. Las Tablas de Hechos son aquellas que tienen el símbolo de color verde, azul y amarillo, las Tablas Dimensionales son las que tienen el símbolo rojo, azul y amarillo.

Sin embargo el usuario puede cambiar esa propuesta de manera manual como se aprecia en la siguiente figura:

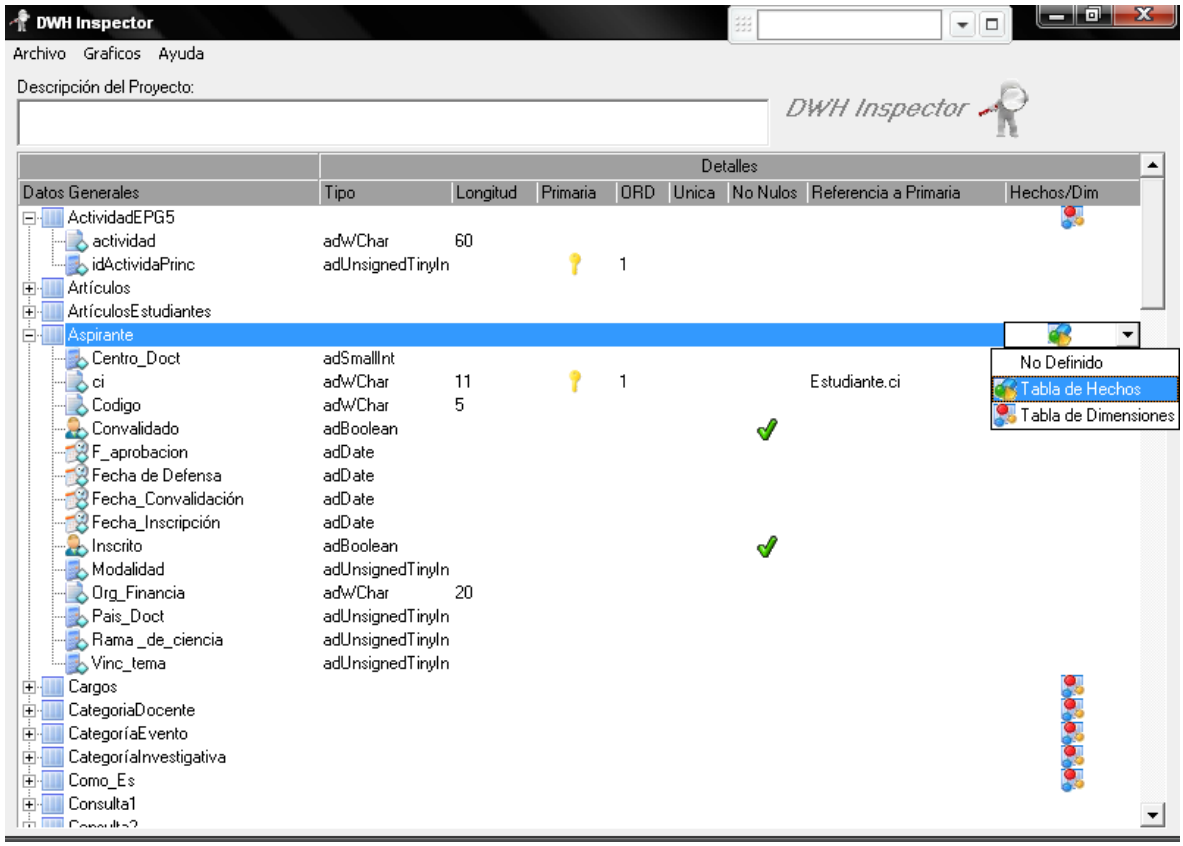


Figura 19. Cambio manual del tipo de tabla.

El usuario puede según su criterio cambiar la propuesta del tipo de tabla hecha por el sistema de una forma visual y cómoda para el mismo, punteando la definición para desplegar un cuadro combinado con las opciones de **No Definido**, **Tabla de Hechos** y **Tabla de Dimensiones** y eligiendo la que desee.

El usuario tiene también la facilidad de ocultar las tablas o vistas que no sean de su interés, el menú contextual que incluye las opciones **Ocultar Tabla** y **Mostrar Tablas Ocultas** (en caso de que existan) se muestra haciendo clic derecho encima de la tabla que se desee ocultar, como se aprecia en la siguiente figura.


Figura 20. Opción de ocultar tabla.

El usuario tiene la facilidad además de ocultar los atributos de tablas determinadas que no sean de su interés, el menú contextual que incluye las opciones **Ocultar Columna** y **Mostrar Columnas**

**ocultas** (en caso de que existan) se muestra haciendo clic derecho encima del atributo que se desee ocultar, como se aprecia en la siguiente figura.

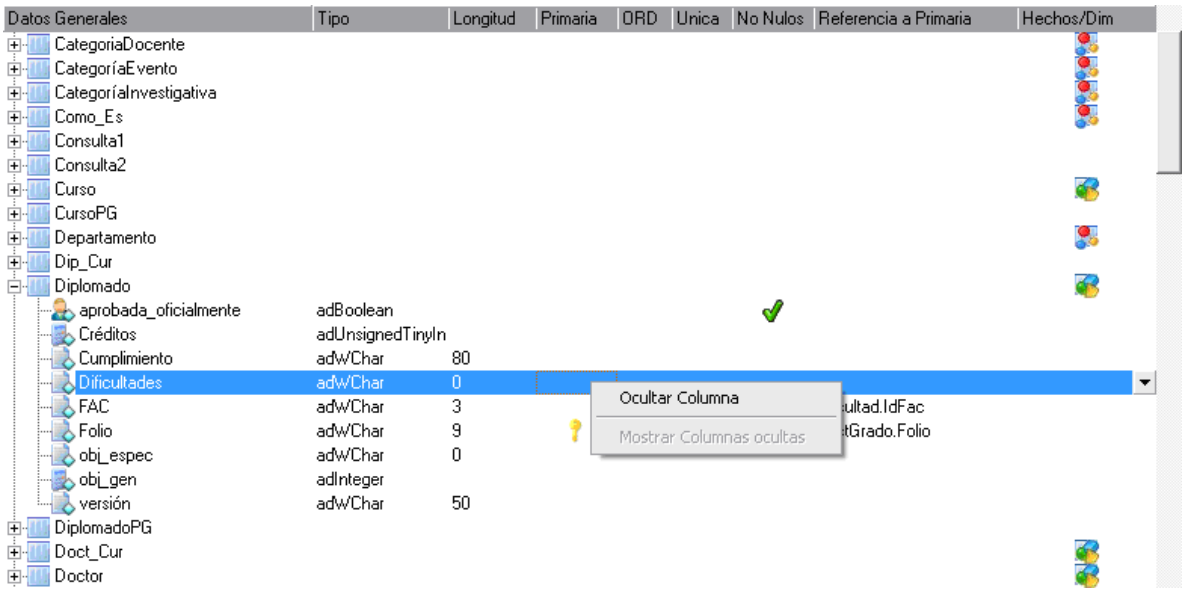


Figura 21. Opción de ocultar columna.

## Mostrar Diagrama Estrella

El usuario podrá mostrar el diagrama de estrella siempre y cuando la tabla que haya seleccionado sea de hechos, usando la opción correspondiente en el menú Gráficos. En el diagrama estrella se detallan visualmente el tipo de dato que contiene cada atributo así como si es una llave primaria.

Como algo específico del CASE, si en la tabla de hechos marcada existe algún tipo de dato fecha, se induce en el diagrama la presencia de la Dimensión Tiempo (Dim_Fecha) característica de los almacenes de datos. El diagrama resultante se aprecia en la siguiente figura.

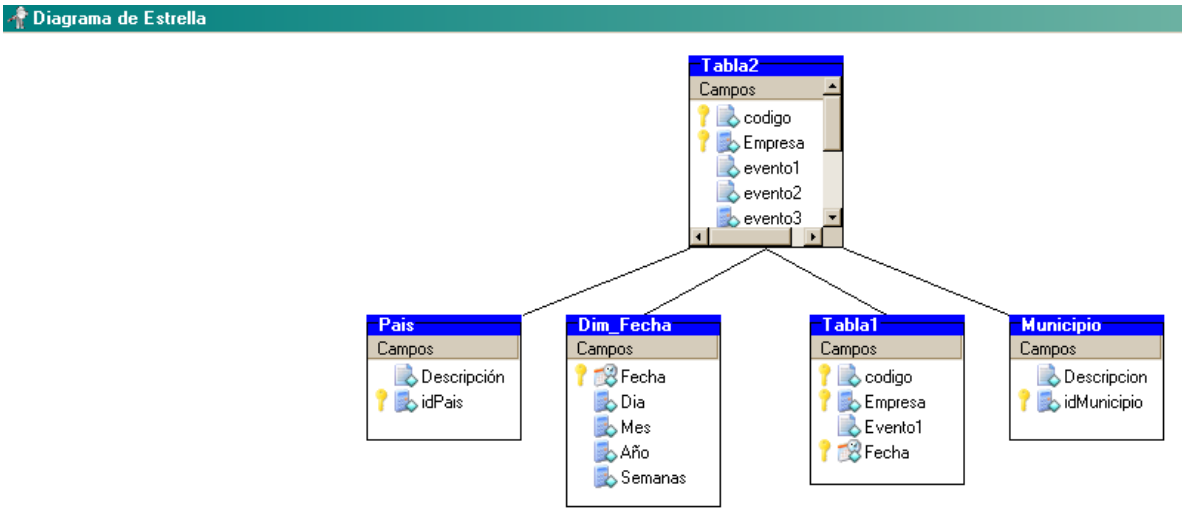


Figura 22. Diagrama estrella propuesto.

## Mostrar Organigramas

Como una facilidad de este sistema se puede mostrar al usuario adicionalmente los gráficos de estrella y copo de nieve en forma de organigrama para ayudar al mismo en la toma de decisiones.

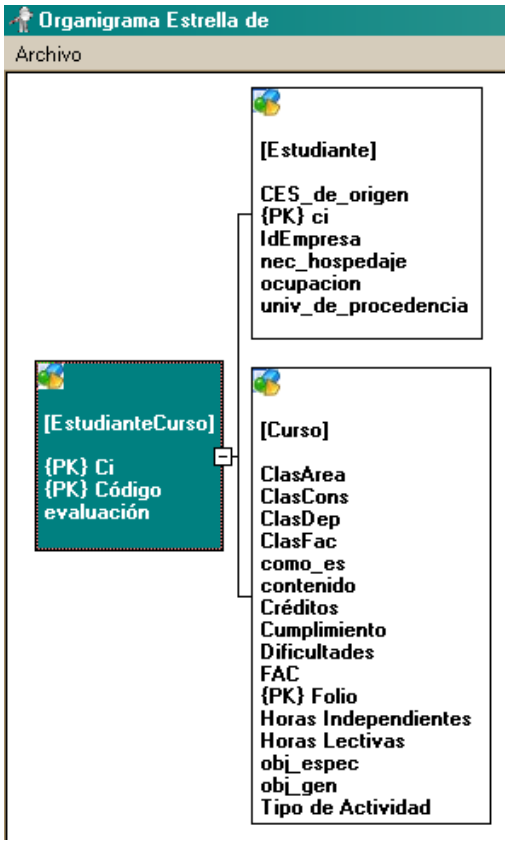


Figura 23. Organigrama Estrella.

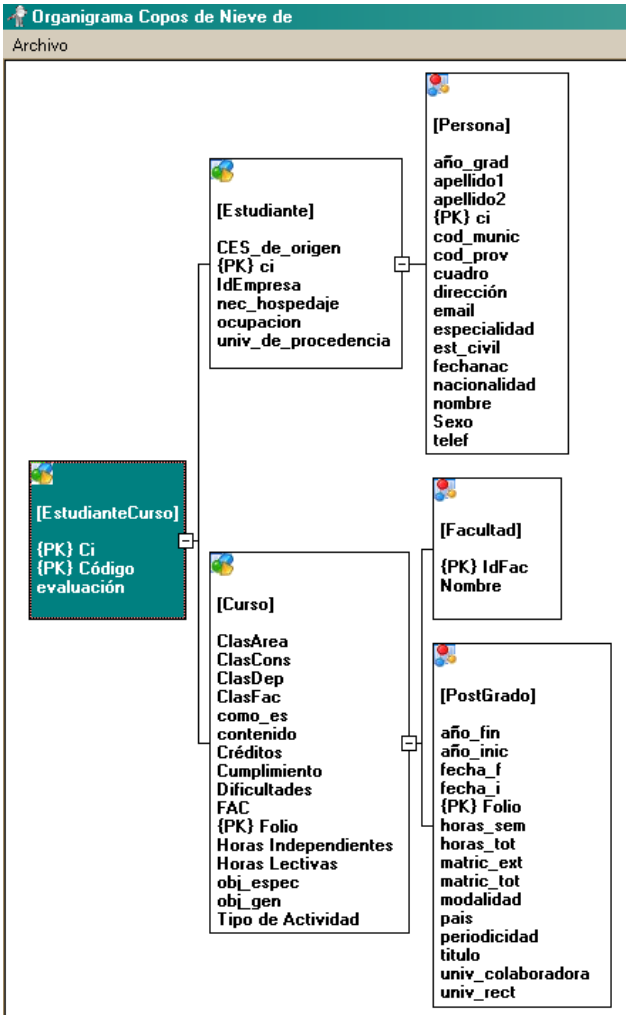


Figura 23. Organigrama Copo de Nieve.

## Guardar Proyecto

El usuario puede salvar su proyecto, haya hecho cambios o no en la propuesta del CASE con respecto al tipo de tablas para el almacén, en un archivo de extensión .DWHI cuyo contenido está en formato XML. En dicha salva van incluidos los datos de las tablas y columnas que el usuario con anterioridad decidió ocultar. Esto se aprecia en la siguiente figura.

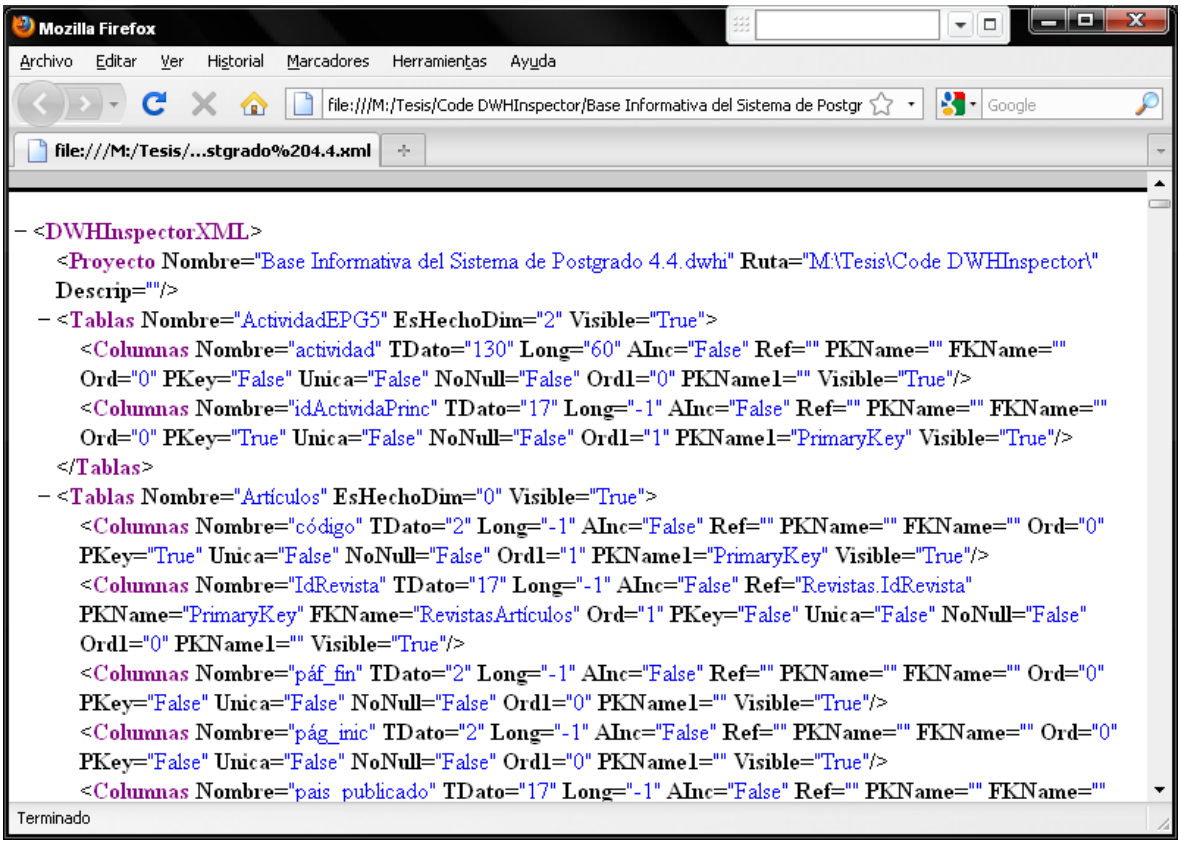


Figura 24. Archivo XML con la salva del proyecto.

## Imprimir la estructura de la Base de Datos original

Otra opción adicional corresponde a la impresión de la estructura de la Base de Datos original, lo cual se logra a partir de la opción **Imprimir** en el menú **Archivo**. El resultado preliminar aparece en pantalla como se aprecia en la siguiente figura.

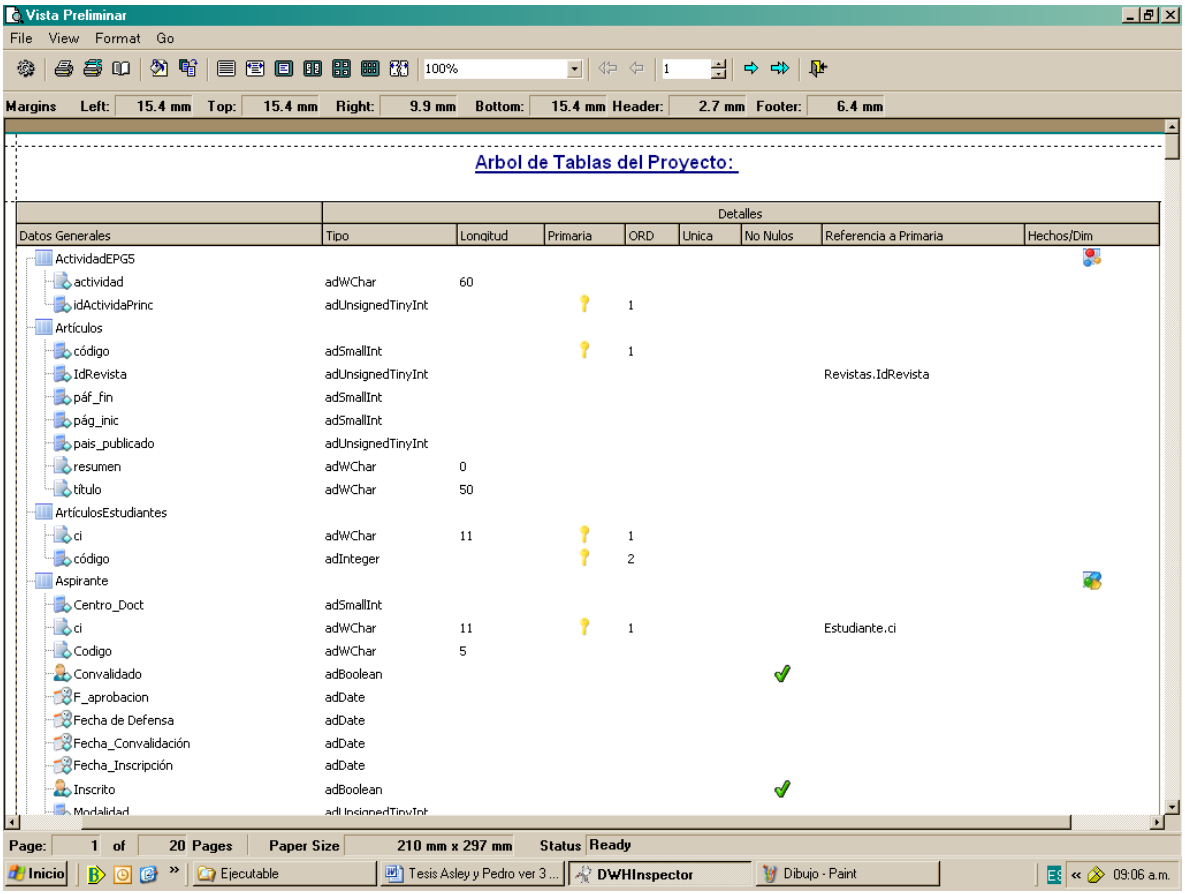


Figura 25. Vista preliminar de la estructura de la Base de Datos analizada en el proyecto.

## 3.3. Conclusiones parciales.

En este capítulo se trató acerca de las características de la aplicación, dicha aplicación cuenta con una ventana principal donde el usuario tiene la posibilidad de importar una base de datos y automáticamente definir en su contenido que tablas son de hechos y cuales son de dimensiones.

En esta ventana el usuario puede ver los atributos de dichas tablas identificando los que sean llaves, fechas o numéricos, cada uno con un color diferente lo que los hace diferentes a los demás atributos, esto puede facilitar mucho el entendimiento del usuario sobre el sistema.

Además la aplicación permite modificar, según el criterio del usuario, la estructura original de la base de datos importada cambiando la definición de dichas tablas, así como ocultando los atributos y tablas que no sean de su interés.

El usuario también puede obtener el diagrama estrella después de seleccionar una tabla de hechos, así como los organigramas de estrella y copo de nieve. Como algo adicional la aplicación también permite imprimir los datos de la base de datos seleccionada.

# Conclusiones.

# Conclusiones.

Tras concluir este trabajo se puede concluir que:

1. Se estudiaron las características propias del modelo dimensional (esquemas estrella o copo de nieve) utilizado mayormente en la confección de Almacenes de Datos y se establecieron sus diferencias con respecto al Modelo Relacional en que se basan la mayoría de los sistemas de información actuales, sobre todo en lo concerniente a la existencia de una gran tabla de hechos central que puede incluir información redundante, pero principalmente con datos mensurables y claves, y rodeada de tablas dimensionales relacionadas a partir de las claves Se determinó como una característica esencial la existencia de las Dimensión Tiempo siempre que aparecieran campos fechas en las tablas de hechos.
2. Se determinaron las pautas de conversión de estructuras relacionales a estructuras dimensionales, basadas en la existencia o no de los tipos de campos numéricos o de fecha en las tablas fuentes, lo que permitió establecer un algoritmo de definición de posibles tablas de hechos y dimensionales, a partir de los metadatos estructurales de los sistemas heredados de determinados formatos relacionales.
3. Se implementó una herramienta CASE (DWH-Inspector) en su primera versión, que aplicando las pautas de conversión anteriormente especificadas, presenta al usuario propuestas de tablas de hechos y dimensionales, a partir de lo cual, se pueda proponer automáticamente una parte del modelo estrella o copo de nieve del almacén de datos sugerido en base a los requerimientos del usuario.
4. Se probó la explotación del CASE DWH-Inspector tomado como caso de estudio la base de datos del Sistema de Control de las Actividades de Postgrado desarrollado por el tutor en años anteriores, probando todas las opciones explicadas del sistema.

# Recomendaciones.

# Recomendaciones.

Una vez concluido el trabajo se recomienda:

- ✓ Divulgar la aplicación para que pueda ser probada por varios usuarios-analistas.
- ✓ Continuar la investigación con el propósito de aumentar las funcionalidades de la aplicación, obteniendo nuevas mejoras en futuras versiones del sistema.

# Referencias Bibliográficas.

# Referencias Bibliográficas.


Anónimo. "Data Wareahousing." Disponible en Internet en: <http://porgramacion.com/bbdd/tutorial/warehouse/>. Consultado el 28/07/2010

Anónimo (1996). Disponible en Internet en: http://www.monografias.com/trabajos14/herramicase/herramicase.shtml Consultado el 28/07/2010.

Anónimo (2002). Disponible en Internet en: http://www.todoexpertos.com/categorias/tecnologia-e-internet/programacion/delphi/respuestas/937115/ventajas-de-delphi Consultado el 28/07/2010.

Anónimo (2005). Disponible en Internet en: <http://es.wikipedia.org/wiki/ADO_%28informatica%29>.Consultado el 30/07/2010.

Anónimo (2007). Disponible en Internet en: <http://dat.etsit.upm.es/~abarbero/curso/xml/xmltutorial.html>. Consulado el 30/07/2010.

Bernabue, I. R. D. (2007). "Data Warehousing: Investigación y Sistematización de Conceptos – Hefesto: Metodología propia para la Construcción de un Data Warehouse". Córdova. **2010**.

Inmon, W. H. (2005). "Building the Data Warehouse". John Wiley & Sons, Inc. 1996.

Kimball, R. (1995). "Fact Tables and Dimension Tables". John Wiley & Sons, Inc. 1990.

Immon, W (1996): "Building the Operational Data Store". John Wiley & Sons, Inc. 1996.

Kimball(1998). "The Data Warehouse Life Cycle Toolkit". Second Edition. Wiley Publishing, Inc. USA.

# Bibliografía.

# Bibliografía.


1. Alter, S. 2002. “Information Systems”. Fourth Edition. Prentice Hall. Pearson Education, Inc. USA.
2. Ambler, S. 2005. “The Elements of UML 2.0 Style”. Cambridge University Press. United Kingdom.
3. Anónimo. MSDN en línea. Por qué XML - Disponible en Internet en http://www.microsoft.com/latam/msdn/articulos/2000/03/art03/#top
4. Boggs, W. & Boggs, M. 2002. “UML with Rational Rose 2002”. SYBEX Inc. USA.
5. Booch, G.; Jacobson, I. & Rumbaugh, J. 2005. “The Unified Modeling Language User Guide”. Second Edition. Addison- Wesley. Pearson Education, Inc. USA.
6. Connolly, T., & Begg, C. 2005. “Sistemas de Bases de Datos. Un enfoque práctico para diseño, implementación y gestión”. Cuarta Edición. Pearson Educación S.A. España.
7. Date, c. J. 2004. “An Introduction to Database Systems”. Eighth Edition. Addison- Wesley. USA.
8. Elmasri, R., Navathe, S. 2007. “Fundamentals of Database Systems”. Fifth Edition. Pearson International / Addison Wesley. USA.
9. Imhoff, Claudia et al. 2003. “Mastering Data Warehouse Design. Relational and Dimensional Techniques”. Wiley Publishing, Inc. USA.
10. Inmon, W.H. 2005. “Building the Data Warehouse”. Fourth Edition. Wiley Publishing, Inc. USA.
11. Inmon, W. H., Teorey, T., et al. 2009. “Database Design Know It All”. Morgan Kaufmann Publishers. USA.
12. Jacobson, I., Booch, G. & Rumbaugh, J. 2000. “El Proceso Unificado de Desarrollo de Software”. Addison- Wesley. España.
13. Kimball, Ralph. 2002. “The Data Warehouse Toolkit”. Second Edition. Wiley Publishing, Inc. USA.
14. Kimball, R. *et al*. 1998. “The Data Warehouse Life Cycle Toolkit”. Second Edition. Wiley Publishing, Inc. USA.
15. Kimball, R., & Caserta, J. 2004. “The Data Warehouse ETL Toolkit”. Wiley Publishing, Inc. USA.
16. Kimball, R., *et al*. 2006. “The Microsoft Data Warehouse Toolkit”. Wiley Publishing, Inc. USA.
17. Montero R. “XML” Disponible en Internet en http://www.ramon.org/xml/articulos/intro_xml-html.htm
18. Muller, R. 1999. “Database Design for Smarties Using UML for Data Modeling”. Academic Press. USA.

19. Poole, John; et al. 2003. “Common Warehouse Metamodel Developer’s Guide”. Wiley Publishing, Inc. USA.
20. Ponniah, Paulraj. 2001. “Data Warehousing Fundamentals. A Comprehensive Guide for IT Professionals”. John Wiley & Sons, Inc. USA.
21. Powell, G. 2006. “Beginning Database Design”. Wiley Publishing, Inc. USA.
22. Pressman, Roger. 2007. “Ingeniería del Software. Un enfoque práctico”. Sexta Edición. Mc Graw Hill. México.
23. Siau, K., Ericsson, J. 2009. “Advanced Principles for Improving Database Design, Systems Modeling, and Software Development”. Information Science Referente. USA.
24. Silvers, Fon. 2008. “Building & Maintaining a Data Warehouse”. CRC Press / Taylor & Francis Group. USA.
25. Simsion, G. C., Witt, G. C. 2005. “Data Modeling Essentials”. Third Edition. Morgan Kaufmann Publishers. USA.
26. Sommerville, Ian. 2007. “Software Engineering”. Eighth Edition. Addison-Wesley. USA.
27. Stackowiak, Robert; et al. 2007. “Oracle Data Warehousing & Business Intelligence Solutions”. Wiley Publishing, Inc. USA.
28. Sumathi, S. 2007. “Fundamentals of Relational Database Management Systems”. Springer. Germany.
29. Teorey, T., Lightstone, S., Nadeau, T. 2006. “Database Modeling & Design: Logical Design”. Fourth Edition. Morgan Kaufmann Publishers. USA.
30. Wang, John. 2006. “Encyclopedia of Data Warehousing and Mining”. Idea Group Referente. USA.
31. Young M. 2000 “XML Step by Step”, Washington, Microsoft Press, p. 12