PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

HERRAMIENTA DE INTELIGENCIA DE NEGOCIOS ORIENTADA A CLIENTES MÓVILES.

RENÉ FRANCISCO ALARCÓN ALALLANA

TESIS DE GRADO
MAGÍSTER EN INGENIERÍA INFORMÁTICA

(Diciembre de 2013)

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

HERRAMIENTA DE INTELIGENCIA DE NEGOCIOS ORIENTADA A CLIENTES MÓVILES.

RENÉ FRANCISCO ALARCÓN ALALLANA

Director de Tesis: **JOSÉ MIGUEL RUBIO**

Programa: **Magíster en Ingeniería Informática**

(Diciembre de 2013)

Resumen

La telefonía móvil en esta última década ha tenido un importante avance con la llegada de los denominados Smartphone. Las personas se dieron cuenta que no solo podían utilizar su teléfono para comunicarse verbalmente, si no que contaban con un sinfín de aplicaciones y utilidades, como las redes sociales, acceso a correos, los chat, entre otros.

Cada vez estos celulares Smartphone y Tablet están más accesibles para las personas convirtiéndose en un producto masivo. Esto representa una ventaja y a la vez una oportunidad para darle un uso extra al celular, como es el caso del desarrollo de las aplicaciones, por lo cual se hace atractivo desarrollar aplicaciones que apoyen las labores del personal de una empresa, para encargados de la dirección del negocio y sobretodo de la toma de decisiones. Esto a través de que estas personas cuenten en forma oportuna con información relevante del negocio en donde quiera que estén, de viaje u en otro lugar fuera de la oficina, información relacionada con las ventas de una empresa, comparación con años anteriores, gráficos. Todo esto a través de la aplicación del celular.

Esta tesis enfoca en diseñar e implementar el uso de la Inteligencia de negocios en los dispositivos móviles, basado en la programación IOS Objective-C (Cocoa) perteneciente a la empresa APPLE orientados a Tablet como IPhone y/o iPad obteniendo resultados eficientes para la toma de decisiones, y realizar un estudio mediante el uso y rendimiento de los cubos OLAP y lo que actualmente la empresa opta por mejoras en la tomas de decisiones desde cualquier lugar, permitiendo agilizar la consulta en base grandes cantidades de datos.

Palabras clave: Inteligencia de Negocios, Smartphone, IOS.

Abstract

Mobile telephony in the last decade has had a major breakthrough with the advent of the "Smartphone". People realized they could not just use their phone to communicate verbally, but it had a lot of applications and utilities such as social networks, access to mails, chat, among others.

Every time Smartphones and Tablets are more accessible to people, becoming in a mass product. This represents an advantage and a great opportunity such as the development of applications that support the work of the staff of a company, give information to the responsible for the business direction and especially are used to support the process of decision making. This allow people to have in a timely manner relevant business information like information related to a company's sales, historical data from previous years or graphic reports, all this wherever they are, even outside the office.

This thesis focuses on designing and implementing the use of business intelligence on mobile devices based on iOS Objective-C programming (Cocoa) belonging to the company Apple and oriented to iPhone and iPad products to obtain efficient results for decision making, the study through the use and performance of OLAP cubes to improve in decision-making from anywhere, allowing the query on large amounts of data.

Keywords: Business Intelligence , Smartphone, iOS.

Índice de Contenido

Índice de Tablas viii

1 Introducción 10

1.1 Introducción 10

1.2 Objetivos del Proyecto 11

1.2.1 Objetivo General 11

1.2.2 Objetivos Específicos 11

2 Inteligencia de Negocios 12

2.1 Introducción 12

2.2 Reseña Histórica 13

2.3 Características 14

2.4 Proceso ETL 15

2.5 Data WareHousing 16

2.5.1 Conceptos básicos asociado a un Data Warehousing 16

2.5.2 Datamart 19

2.5.3 Diseño de un Data Warehouse 21

2.5.4 Diseño Conceptual 22

2.5.5 Diseño Lógico 22

2.5.6 Esquema Estrella 23

2.5.7 Esquema Copo de Nieve 24

2.5.8 Esquema de Constelación 25

2.5.9 OLTP v/s DWS 26

2.6 Consultas y Reportes 27

2.7 Microsoft como herramienta de Inteligencia de Negocios 28

2.8 SQL 2008 como herramienta de Inteligencia de Negocio. 28

2.9 Ediciones disponibles SQL Server 2008 29

2.10 Requerimiento de Hardware 30

2.11 Inteligencia de Negocios en proyectos móviles 30

3 Herramientas de desarrollo 32

3.1 Xcode 32
3.2 Objective C 34
3.2.1 Historia 34
3.2.2 Sintaxis 35
3.3 Java 36
3.3.1 Introducción 36
3.3.2 Introducción a los Servlet 37
3.3.3 Metodología de desarrollo de un Servlet 38
3.3.4 Característica de los Servlet 39
3.3.5 Equivalencia entre CGI y Servlet 40
3.3.6 Ciclo de vida de un servlet 42
3.3.7 Obteniendo datos del cliente 44
3.3.8 El trabajo con las cabeceras 44
3.3.9 Códigos de Estados 46
3.4 IOS 47
3.4.1 Introducción 47
3.4.2 Smartphone 50
4 Caso de Estudio 52
4.1 Caso de estudio del proyecto 52
4.2 Cubo análisis OLAP 53
4.3 Tabla de hechos 54
4.3.1 Medidas tabla de hechos 54
4.3.2 Cardinalidad tabla de hechos 54
4.3.3 Granularidad 55
4.3.4 Agregación 55
4.4 Arquitectura 56
5 Discusión de Resultados 57
5.1 Análisis de datos 57
Conclusiones y Trabajo Futuro 69

Índice de Figuras

Figura 1 - Modelo Inteligencia de Negocios. Fuente: Elaboración propia. 12
Figura 2 - Proceso ETL. Fuente: Elaboración propia. 15
Figura 3 - Proceso Data Warehouse. Fuente: Elaboración propia. 18
Figura 4 - Modelo Top-down. Fuente: Elaboración propia. 20
Figura 5 - Modelo Bottom-up. Fuente: Elaboración propia. 20
Figura 6 - Proceso de diseño de un Data Warehouse. Fuente: Elaboración propia. 21
Figura 7 - Esquema Estrella. Fuente: Elaboración propia. 23
Figura 8 - Esquema Copo de Nieve. Fuente: Elaboración propia. 24
Figura 9 - Esquema de Constelación. Fuente: Elaboración propia. 25
Figura 10 - OLTP v/s Data Warehouse. Fuente: Elaboración propia. 26
Figura 11 - Proceso Consulta y Reportes. Fuente: Elaboración propia. 27
Figura 12 - Componentes básicos de SQL Server 2008. 29
Figura 13 - Proyecto Inteligencia de Negocios en iPhone. 31
Figura 14 - Herramienta Xcode. 33
Figura 15 – Jerarquía. Fuente: Elaboración propia. 38
Figura 16 - Peticiones 1. Fuente: Elaboración propia. 42
Figura 17 - Peticiones 2. Fuente: Elaboración propia. 43
Figura 18 - Base de datos Bata S.A. Fuente: Elaboración propia. 52
Figura 19 - Ejemplo pasó 3. Fuente: Elaboración propia. 53
Figura 20 - Arquitectura proyecto móvil. Fuente: Elaboración propia. 56
Figura 21 - Modelo de tabla Estrella. Fuente: Elaboración propia (Generado en SQL Server 2008) 57
Figura 22 - Modelo de tabla Copo de nieve. Fuente: Elaboración propia (Generado en SQL Server 2008). 58
Figura 23 - Proceso modelo Estrella. Fuente: Elaboración propia (Generado en SQL Server 2008). 59
Figura 24 - Proceso modelo Copo de nieve. Fuente: Elaboración propia (Generado en SQL Server 2008). 59
Figura 25 - Consulta generada a través de Analysis Service modelo Estrella. Fuente: Elaboración propia. 60
Figura 26 - Consulta generada a través de Analysis Service modelo Copo de nieve. Fuente: Elaboración propia. 61
Figura 27 - Consulta generada a través de Excel modelo Estrella. Fuente: Elaboración propia. 61
Figura 28 - Consulta generada a través de Excel modelo Copo de nieve. Fuente: Elaboración propia. 62
Figura 29 - Configuración IIS para Analysis service. Fuente: Elaboración propia (Generado en SQL Server 2008) 64
Figura 30 – Pantalla de inicio de la aplicación en IPad. Fuente: Elaboración propia. 65
Figura 31 – Ventas totales por año (Estrella). Fuente: Elaboración propia. 66
Figura 32 – Ventas totales por año (Nieve). Fuente: Elaboración propia. 66
Figura 33 – Unidades ventas por semestre (Estrella). Fuente: Elaboración propia. 67
Figura 34 – Unidades ventas por semestre (Nieve). Fuente: Elaboración propia. 67

Figura 35 – Unidades vendidas por año y categoría (Estrella). Fuente: Elaboración propia. 68
Figura 36 – Unidades vendidas por año y categoría (Nieve). Fuente: Elaboración propia..... 68

Índice de Tablas

Tabla 1 - Ediciones disponibles en SQL Server 2008 30
Tabla 2 – Requerimiento mínimo de Hardware..... 30
Tabla 3 - Cuadro comparativo entre CGI y Servlet 41
Tabla 4 - Aplicaciones básicas de un iPhone..... 48
Tabla 5 - Cardinalidad de una tabla x..... 54
Tabla 6 - Ejemplo de conexión con OLAP4J. 64

Lista de Abreviaturas o Siglas

- ✓ AD HOC: se refiere a una solución elaborada específicamente para un problema o fin preciso.
- ✓ BI: Inteligencia de Negocios (Business Intelligence).
- ✓ DSS: Sistema de soporte a las decisiones, permite automatizar los procesos de tomas de decisiones.
- ✓ DW: DataWarehouse.
- ✓ EIS: Sistema de información ejecutiva, herramienta que permite monitorear el estado de las variables de aérea o unidad de la empresa a partir de información externa o interna de la misma.
- ✓ ETL: Extracción (Extraction), Transformación (Transformation), Carga (Load).
- ✓ J2EE: Java 2 Platform Enterprise Edition o informarmente Java Empresarial.
- ✓ KPI: Indicadores Clave de Desempeño (Key Performance Indicators), miden el nivel del desempeño. de un proceso.
- ✓ LLVM: Low Lever Virtal Machine o Máquina virtual de bajo nivel.
- ✓ SGBD: Sistemas de gestión de bases de datos (data base management system, abreviado DBMS).

CAPÍTULO 1

1 Introducción

1.1 Introducción

Hoy en día la tecnología móvil presenta un avance importante en la forma en que se comunican las personas, ya no es sólo llamar de un lugar a otro, o una comunicación uno a uno, sino más bien a un conjunto de personas en tiempo real comunicándose o compartiendo información. Con la llegada de internet a los dispositivos móviles las empresas han percibido una nueva forma de gestionar la información.

La Inteligencia de Negocios (BI) es el proceso de analizar los datos acumulados en la empresa y extraer un conocimiento de ellos. Se incluyen las bases de datos de clientes, información de la cadena de suministro, ventas personales y cualquier actividad de marketing o fuente de información relevante para la empresa entre otros.

BI apoya a los tomadores de decisiones con la información correcta, en el momento y lugar correcto, lo que les permite tomar mejores decisiones de negocios. La información adecuada en el lugar y momento adecuado incrementa efectividad de cualquier empresa.

La fusión entre Business Intelligence y la telefonía móvil, permite distribuir datos relevantes en cualquier momento y lugar, con lo que los profesionales encuentran muy cómodo el uso de teléfonos inteligentes o Smartphone, no sólo con fines de comunicación sino también como un medio para mantenerse al día con la información del estado del negocio.

Esta tesis enfoca en diseñar e implementar el uso de la Inteligencia de negocios en los dispositivos móviles, basado en la programación IOS Objective-C (Cocoa) perteneciente a la empresa APPLE orientados a Tablet como IPhone y/o iPad obteniendo resultados eficientes para la toma de decisiones, y realizar un estudio mediante el uso y rendimiento de los cubos OLAP y lo que actualmente la empresa ocupa para sus tomas de decisiones, que permitan agilizar la consulta de grandes cantidades de datos.

1.2 Objetivos del Proyecto

1.2.1 Objetivo General

El objetivo de la tesis es la construcción de una aplicación móvil que facilite la toma de decisiones de la Gerencia de Ventas de una compañía, mediante la utilización de Inteligencia de Negocios. Esto permitirá al área de negocio aplicar estrategias operativas para logar su objetivo.

1.2.2 Objetivos Específicos

- ☑ Implementación de un sistema móvil que permita obtener datos específicos a través de la utilización de cubos.
- ☑ Creación de cubos mediante Analisys Service para el estudio e interpretación de datos.
- ☑ Pruebas de tiempo de respuesta y creación de gráficos comparativos de los resultados de esquemas de copo de nieve v/s estrella.

CAPÍTULO 2

2 Inteligencia de Negocios

2.1 Introducción

La Inteligencia de Negocios se define como un conjunto de estrategias y herramientas enfocadas a la administración y creación de conocimiento mediante el análisis de datos existentes en una organización o empresa [3]. Esta alternativa tecnológica permite manejar la información requerida por alguna empresa u organización sirviendo de apoyo a la toma de decisiones estratégicas, que comprende desde la extracción de los datos de los sistemas existentes hasta la explotación de la información por las herramientas de análisis de datos.

Por otro lado la Inteligencia de Negocios se puede resumir como el manejo inteligente y creativo de los datos disponibles relativo a una compañía o institución ya sea privado o gubernamental en la transformación de estos datos en información.

La Inteligencia de Negocios junto con la información se transformará en tomas de decisiones correctas, esta información permitirá mejorar y/o crear información que proporcione respuestas. Esta nueva información permitirá a las personas tomar decisiones.

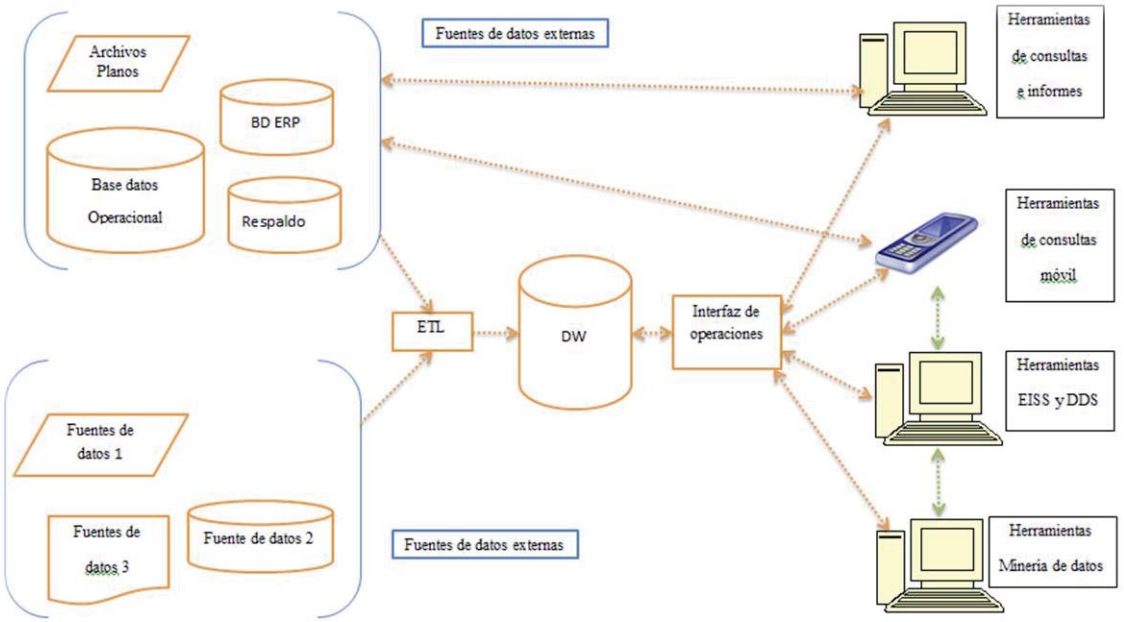


Figura 1 - Modelo Inteligencia de Negocios. Fuente: Elaboración propia.

2.2 Reseña Histórica

En un tiempo, las organizaciones dependían de sus departamentos de sistemas de información para proporcionarles reportes estándar y personalizados. Esto ocurrió en los días de los mainframes y minicomputadoras, cuando la mayoría de los usuarios no tenía acceso directo a las computadoras. Sin embargo, esto comenzó a cambiar en los años 70 cuando los sistemas basados en servidores se convirtieron en la moda.

Aun así, estos sistemas eran usados principalmente para transacciones de negocios y sus capacidades de realizar reportes se limitaban a un número predefinido de ellos. Los sistemas de información se sobrecargaban y los usuarios tenían que esperar por días o semanas para obtener sus informes en caso que requirieran reportes distintos a los estándares disponibles [2].

Con el paso del tiempo, fueron desarrollados los Sistemas de Información Ejecutiva (EIS, por sus siglas en inglés), los cuales fueron adaptados para apoyar a las necesidades de ejecutivos y administradores. Con la entrada de la PC, y de computadoras en red, las herramientas de BI proveyeron a los usuarios de la tecnología para crear sus propias rutinas básicas y reportes personalizados.
Hitos históricos de la Inteligencia de Negocios:

- ✓ 1969: Creación del concepto de base de datos (Codd) [6].
- ✓ 1970: Desarrollo de las primeras bases de datos y las primeras aplicaciones empresariales (SAP, JD Edwards, Siebel, PeopleSoft). Estas aplicaciones permitieron realizar "data entry" en los sistemas, aumentando la información disponible, pero no fueron capaces de ofrecer un acceso rápido y fácil a dicha información [6].
- ✓ 1980: Creación del concepto Datawarehouse (Ralph Kimball, Bill Inmon), y aparición de los primeros sistemas de reporting. A pesar de todo, seguía siendo complicado y funcionalmente pobre. Existían relativamente potentes sistemas de bases de datos, pero no había aplicaciones que facilitasen su explotación [6].
- ✓ 1989: Introducción del término Inteligencia de Negocio (Howard Dresner) [7].
- ✓ 1990: Inteligencia de Negocio 1.0. Proliferación de múltiples aplicaciones BI. Estos proveedores resultaban caros, pero facilitaron el acceso a la información, y en cierto modo agravaron el problema que pretendían resolver [8].
- ✓ 2000: Inteligencia de Negocio 2.0. Consolidación de las aplicaciones BI en unas pocas plataformas de Inteligencia de Negocios (Oracle, SAP, IBM, Microsoft). A parte de la información estructurada, se empieza a considerar otro tipo de información y documentos no estructurados.

2.3 Características

La solución es un sistema:

- ✓ Para soporte de toma de decisiones (nivel gerencial).
- ✓ Capacidad de análisis de alcance empresarial global.
  - o Integración y análisis de la información desde fuentes de datos heterogéneas.
- ✓ Plataforma integrada
  - o Herramientas ETL de gestión, administración y carga, el almacenamiento de datos y las funciones relacionadas con informes, servicios.
  - o Procedimientos analíticos integrados (OLAP).
  - o Minería de datos (Uso de métodos inteligentes para extraer conocimiento).
- ✓ Servicio de análisis
  - o Entorno en tiempo real
- ✓ Indicadores claves de desempeño
- ✓ Servicio de reporte
  - o Creación, administración y visualización e informes.
  - o Motor para alojar y procesar informes.
- ✓ Inteligencia de negocio provee soluciones a nivel empresarial que permiten a los tomadores de decisiones transformar información clave de negocio en acciones concretas traduciéndose en beneficios tangibles
  - o Reducción de costos
  - o Mayor rentabilidad
  - o Mejores relaciones comerciales

2.4 Proceso ETL

ETL se define como Extracción (Extraction), Transformación (Transformation), Carga (Load). Realiza la extracción desde las fuentes de datos (internas o externas), transformación (limpieza, consolidación,..) y la carga del DW (carga inicial que considera ordenación, agregación, etc; y refresco del almacén u operación periódica que propaga los cambios de las fuentes al almacén) [3].

El ETL puede ser una herramienta existente en el mercado o programas diseñados especialmente para el proyecto DW.

Cada DW debe tener su propio ETL y es responsabilidad del equipo de desarrollo del DW construir un sistema ETL.

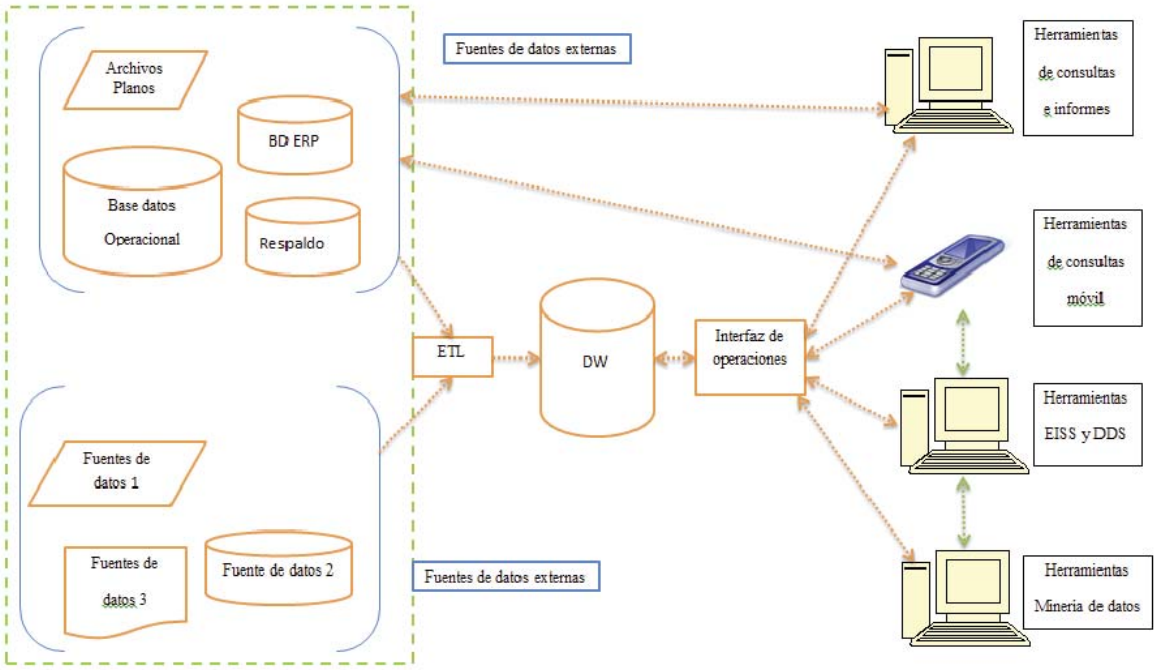


Figura 2 - Proceso ETL. Fuente: Elaboración propia.

2.5 Data WareHousing

2.5.1 Conceptos básicos asociado a un Data Warehousing

El data warehousing consiste en un conjunto de herramientas que permite construir un data warehouse. Un Datawarehouse es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta. La creación de un datawarehouse representa en la mayoría de las ocasiones el primer paso, desde el punto de vista técnico, para implantar una solución completa y fiable de inteligencia de negocios.

La ventaja principal de este tipo de bases de datos radica en las estructuras en las que se almacena la información (modelos de tablas en estrella, en copo de nieve, cubos relacionales... etc). Este tipo de persistencia de la información es homogénea y fiable, y permite la consulta y el tratamiento jerarquizado de la misma (siempre en un entorno diferente a los sistemas operacionales).

El término Datawarehouse fue acuñado por primera vez por Bill Inmon [6], y se traduce literalmente como almacén de datos. No obstante, y como cabe suponer, es mucho más que eso. Según definió el propio Bill Inmon, un datawarehouse se caracteriza por ser:

- ✓ **Integrado:** los datos almacenados en el datawarehouse deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas. La información suele estructurarse también en distintos niveles de detalle para adecuarse a las distintas necesidades de los usuarios.
- ✓ **Temático:** sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla del datawarehouse. De esta forma, las peticiones de información sobre clientes serán más fáciles de responder dado que toda la información reside en el mismo lugar.
- ✓ **Histórico:** el tiempo es parte implícita de la información contenida en un datawarehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información almacenada en el datawarehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el datawarehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- ✓ **No volátil:** el almacén de información de un datawarehouse existe para ser leído, pero no modificado. La información es por tanto permanente, significando la actualización del datawarehouse la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Otra característica del datawarehouse es que contiene metadatos, es decir, datos sobre los datos. Los metadatos permiten saber la procedencia de la información, su periodicidad de refresco, su fiabilidad, forma de cálculo... etc.
Los metadatos serán los que permiten simplificar y automatizar la obtención de la información desde los sistemas operacionales a los sistemas informacionales.
Los objetivos que deben cumplir los metadatos, según el colectivo al que va dirigido, son:

- ✓ **Dar soporte al usuario final**, ayudándole a acceder al datawarehouse con su propio lenguaje de negocio, indicando qué información hay y qué significado tiene. Ayudar a construir consultas, informes y análisis, mediante herramientas de Business Intelligence como DSS, EIS o CMI.
- ✓ **Dar soporte a los responsables técnicos del datawarehouse en aspectos de auditoría**, gestión de la información histórica, administración del datawarehouse, elaboración de programas de extracción de la información, especificación de las interfaces para la realimentación a los sistemas operacionales de los resultados obtenidos... etc.

Principales ventajas de un datawarehouse:

- ✓ Proporciona una herramienta para la toma de decisiones en cualquier área funcional, basándose en información integrada y global del negocio.
- ✓ Facilita la aplicación; obteniendo un valor añadido para el negocio de dicha información.
- ✓ Proporciona la capacidad de aprender de los datos del pasado y de predecir situaciones futuras en diversos escenarios.
- ✓ Simplifica dentro de la empresa la implantación de sistemas de gestión integral de la relación con el cliente.

Supone una optimización tecnológica y económica en entornos de Centro de Información, estadística o de generación de informes con retornos de la inversión espectaculares.

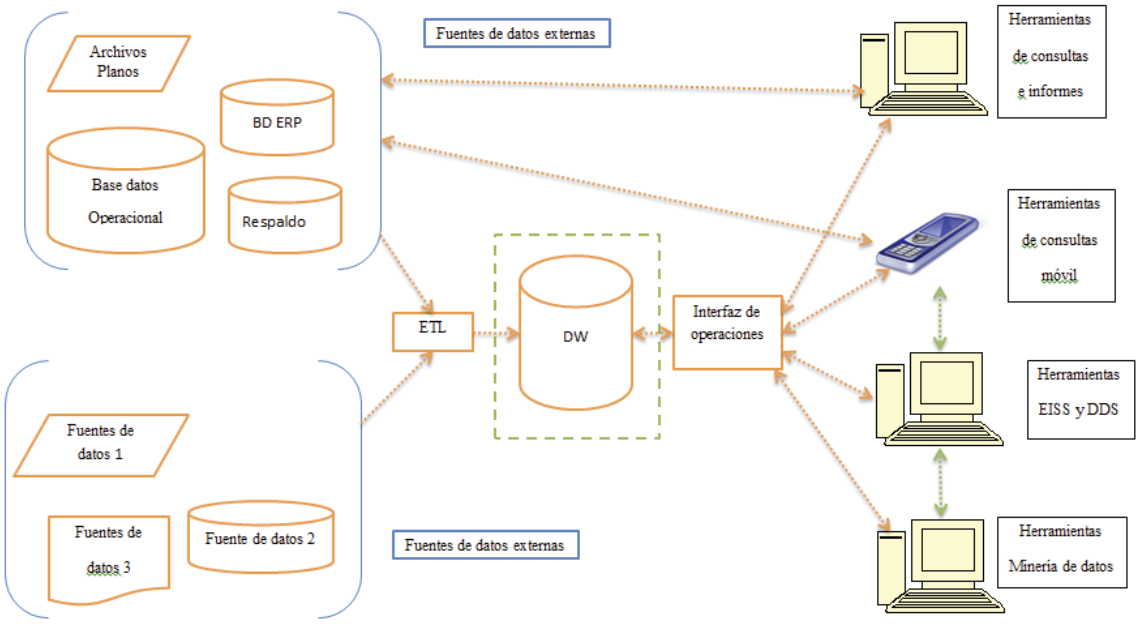


Figura 3 - Proceso Data Warehouse. **Fuente: Elaboración propia.**

2.5.2 **Datamart**

Un Datamart es una base de datos departamental, especializada en el almacenamiento de los datos de un área de negocio específica. Se caracteriza por disponer la estructura óptima de datos para analizar la información al detalle desde todas las perspectivas que afecten a los procesos de dicho departamento. Un datamart puede ser alimentado desde los datos de un datawarehouse, o integrar por sí mismo un compendio de distintas fuentes de información.

Por tanto, para crear el datamart de un área funcional de la empresa es preciso encontrar una estructura óptima para el análisis de su información, estructura que puede estar montada sobre una base de datos OLTP, como el propio datawarehouse, o sobre una base de datos OLAP. La designación de una u otra dependerá de los datos, los requisitos y las características específicas de cada departamento. De esta forma se pueden plantear dos tipos de datamarts:

- ✓ **Datamart OLAP:** Se basan en los populares cubos OLAP, que se construyen agregando, según los requisitos de cada área o departamento, las dimensiones y los indicadores necesarios de cada cubo relacional. El modo de creación, explotación y mantenimiento de los cubos OLAP es muy heterogéneo, en función de la herramienta final que se utilice.
- ✓ **Datamart OLTP:** Pueden basarse en un simple extracto del datawarehouse, no obstante, lo común es introducir mejoras en su rendimiento (las agregaciones y los filtrados suelen ser las operaciones más usuales) aprovechando las características particulares de cada área de la empresa. Las estructuras más comunes en este sentido son las tablas report, que vienen a ser fact-tables reducidas (que agregan las dimensiones oportunas), y las vistas materializadas, que se construyen con la misma estructura que las anteriores, pero con el objetivo de explotar la re-escritura de queries (aunque sólo es posibles en algunos SGBD avanzados, como Oracle).

Los datamarts que están dotados con estas estructuras óptimas de análisis presentan las siguientes ventajas:

- ✓ Poco volumen de datos
- ✓ Mayor rapidez de consulta
- ✓ Consultas SQL y/o MDX sencillas
- ✓ Validación directa de la información
- ✓ Facilidad para historial de los datos

Tipos de diseños:

- ✓ **Top-Down:** Los requerimientos de usuarios en diferentes niveles organizacionales son mezclados antes de que el proceso de diseño comience, y un esquema para el data warehouse completo es construido. Luego, los data marts son obtenidos de acuerdo a las características particulares de cada área de negocio o proceso [3].

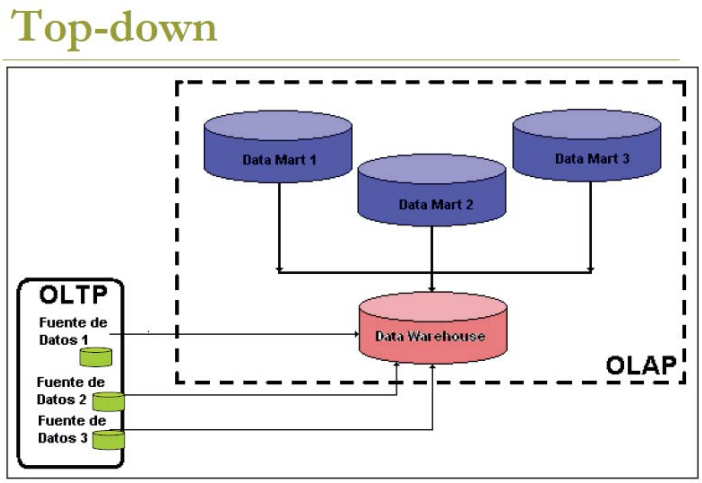


Figura 4 - Modelo Top-down. Fuente: Elaboración propia.

- ✓ **Bottom-up:** Un esquema separado es construido por cada data mart, tomando en cuenta los requerimientos de los usuarios responsables de la toma de decisiones, para los correspondientes procesos o áreas de negocios específicos [3].

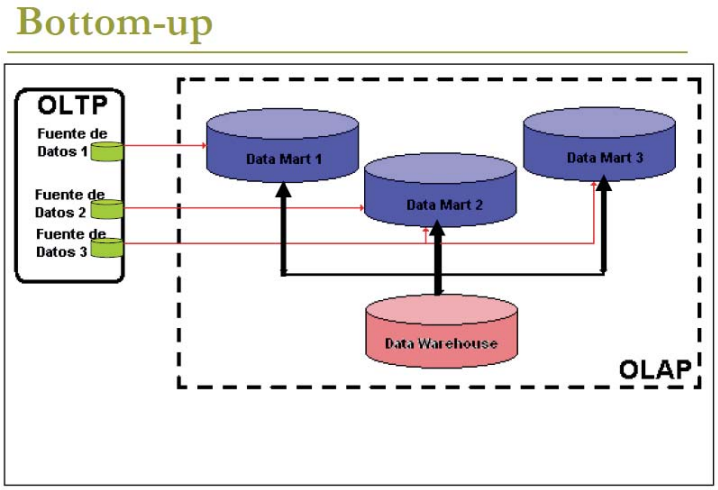


Figura 5 - Modelo Bottom-up. Fuente: Elaboración propia.

2.5.3 **Diseño de un Data Warehouse**

Los sistemas de Data Warehousing han sido objeto de variados trabajos de investigación en la última década. Los trabajos comprenden diferentes áreas y diferentes enfoques. Sus marcadas diferencias con los sistemas operacionales provocaron el estudio de nuevas técnicas y metodologías de diseño, como en los sistemas de bases de datos tradicionales, el proceso de diseño del DW puede dividirse en tres etapas secuenciales: diseño conceptual, diseño lógico y diseño físico.

En la Figura 6 se muestran las etapas con sus respectivas entradas y salidas de información.

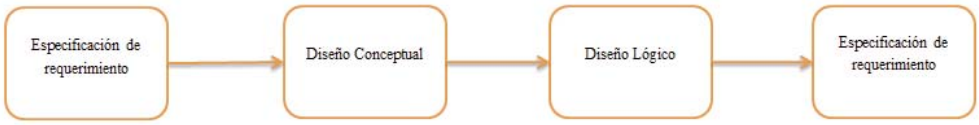


Figura 6 - Proceso de diseño de un Data Warehouse. Fuente: Elaboración propia.

2.5.4 **Diseño Conceptual**

El diseño conceptual tiene por objetivo la construcción de una descripción abstracta y completa del problema. Comienza con el análisis de requerimientos de los usuarios y de reglas de negocio, y finaliza con la construcción de un esquema conceptual expresado en términos de un modelo conceptual. Los trabajos existentes en diseño conceptual para DWs corresponden fundamentalmente a modelos de datos. Dichos modelos de datos son modelos multidimensionales que expresan la realidad en términos de dimensiones y medidas.

En el enfoque basado en requerimientos se analizan los requerimientos de los usuarios, y se identifican en ellos los hechos, dimensiones y medidas relevantes. La realidad se modela como un conjunto de cubos multidimensionales, que se obtienen a partir de los hechos, dimensiones y medidas identificados. Para relevar los requerimientos se depende de la experiencia del diseñador y de entrevistas a los usuarios; es una fase poco automatizable.

En el enfoque basado en las bases fuentes se construyen cubos multidimensionales transformando un esquema conceptual de las bases fuentes. Como modelo conceptual de las fuentes, en general se utiliza el modelo E/R. Las diferentes metodologías comienzan por identificar en el esquema fuente los posibles hechos relevantes para la toma de decisiones. A partir de los hechos identificados navegan por las entidades y relaciones construyendo las jerarquías de las dimensiones [2].

2.5.5 **Diseño Lógico**

La etapa de diseño lógico toma como entrada un esquema conceptual y genera un esquema lógico relacional o multidimensional. La dificultad principal es encontrar un esquema lógico que satisfaga no sólo los requerimientos funcionales de información, sino también requerimientos de performance en la realización de consultas complejas de análisis de datos. Esto tiene particular impacto en el caso de usarse bases relacionales, ya que las consultas de análisis de datos incluyen operaciones muy costosas para DBMS relacionales.

Algunas metodologías parten de un esquema conceptual y proveen mecanismos para generar un esquema lógico a partir de él, otras metodologías pasan por alto el modelado conceptual y construyen el esquema lógico a partir de los requerimientos y/o las bases fuentes.

El resultado de esta etapa es la especificación formal de un esquema lógico para el DW. Los modelos propuestos incluyen materialización de vistas, modelos específicos basados en el modelo relacional y optimizados para consultas OLAP, e implementaciones multidimensionales, en general propietarias de los manejadores.

En el enfoque basado en vistas, el DW se ve como un conjunto de vistas materializadas de las bases fuentes. Estos trabajos no se centran en la representación de conceptos multidimensionales, como dimensiones y medidas, sino en materializar algunas vistas para lograr performance en un conjunto dado de consultas [2].

Otro enfoque consiste en definir estructuras (dentro del modelo relacional) que optimicen las consultas que se realizarán al DW. Dichas consultas contienen gran número de joins y sumarizaciones que degradan la performance del sistema.

En [6], Kimball propone el modelo estrella (star), que consiste de una gran tabla central conteniendo información sobre los hechos, y tablas más pequeñas (relacionadas a la tabla de hechos) con información sobre las dimensiones. Kimball propone lineamientos prácticos para construir un esquema estrella, pero no presenta una metodología general. Varios trabajos de modelización conceptual generan esquemas estrella a partir de esquemas conceptuales.

En [7], Kortink y Moody analizan varios modelos (flat, terraced, star, constellation, galaxy, snowflake, starcluster) variantes del modelo estrella, y proponen una metodología para construirlos a partir de un esquema E/R de las fuentes.

Un enfoque diferente consiste en aplicar sucesivas transformaciones a una base fuente integrada hasta obtener el esquema lógico deseado para el DW. En [8] y [9], Marotta define un conjunto de transformaciones de esquemas que permite llegar a diferentes estilos de diseño (star, snowflake,..) según los requerimientos de performance y almacenamiento del problema. El trabajo incluye un conjunto de reglas y estrategias que facilitan la elección de las transformaciones a aplicar. La aplicación de las transformaciones deja una traza de diseño que permite repercutir en el DW los cambios ocurridos en la fuente, y de esta manera manejar la evolución de la fuente.

2.5.6 **Esquema 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. En la siguiente figura se puede apreciar un esquema en estrella estándar:

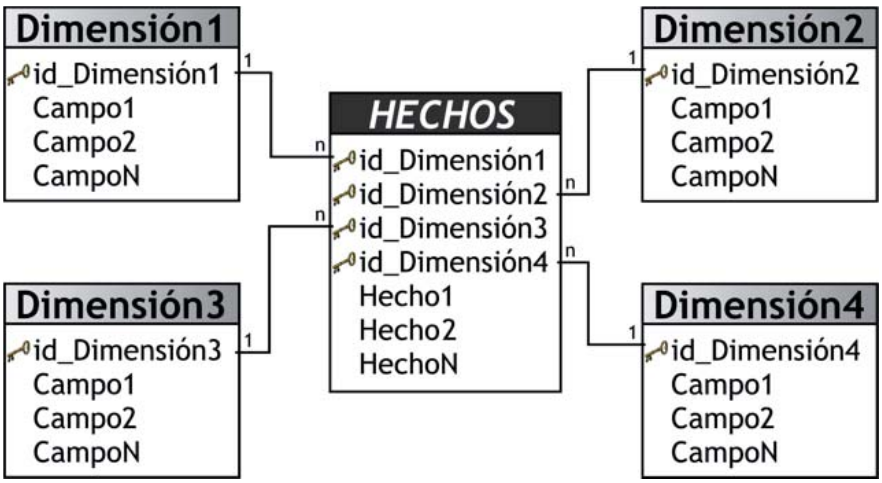


Figura 7 - Esquema Estrella. Fuente: Elaboración propia.

2.5.7 **Esquema Copo de Nieve**

Este esquema representa una extensión del modelo en estrella cuando las tablas de dimensiones se organizan en jerarquías de dimensiones.

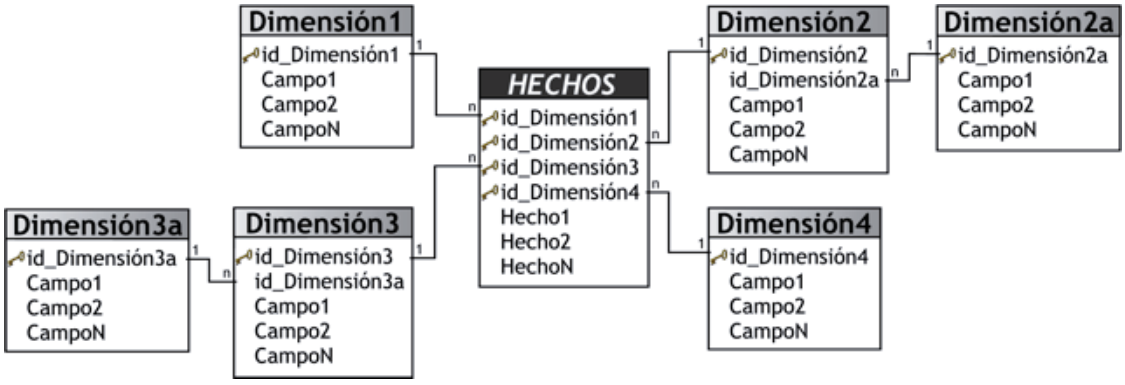


Figura 8 - Esquema Copo de Nieve. Fuente: Elaboración propia.

Como se puede apreciar en la figura anterior, existe una tabla de hechos central que está relacionada con una o más tablas de dimensiones, quienes a su vez pueden estar relacionadas o no con una o más tablas de dimensiones. Este modelo es más cercano a un modelo de entidad relación, que al modelo en estrella, debido a que sus tablas de dimensiones están normalizadas.

Una de los motivos principales de utilizar este tipo de modelo, es la posibilidad de segregar los datos de las tablas de dimensiones y proveer un esquema que sustente los requerimientos de diseño. Otra razón es que es muy flexible y puede implementarse después de que se haya desarrollado un esquema en estrella.

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 tablas de dimensiones están normalizadas, por lo que requiere menos esfuerzo de diseño.
- ✓ Puede desarrollar clases de jerarquías fuera de las tablas de 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 tablas de dimensiones, cada una de ellas con varias jerarquías, se creará un número de tablas bastante considerable, que pueden llegar al punto de ser inmanejables.
- ✓ Al existir muchas uniones y relaciones entre tablas, el desempeño puede verse reducido.

La existencia de las diferentes jerarquías de dimensiones debe estar bien fundamentada, ya que de otro modo las consultas demorarán más tiempo en devolver los resultados, debido a que se deben realizar las uniones entre las tablas.

2.5.8 **Esquema de Constelación**

Este modelo está compuesto por una serie de esquemas en estrella, y tal como se puede apreciar en la siguiente figura, está formado por una tabla de hechos principal ("HECHOS_A") y por una o más tablas de hechos auxiliares ("HECHOS_B"), las cuales pueden ser sumarizaciones 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 sólo algunas de las tablas de dimensiones asignadas a la tabla de hechos principal, y también pueden hacerlo con nuevas tablas de dimensiones.

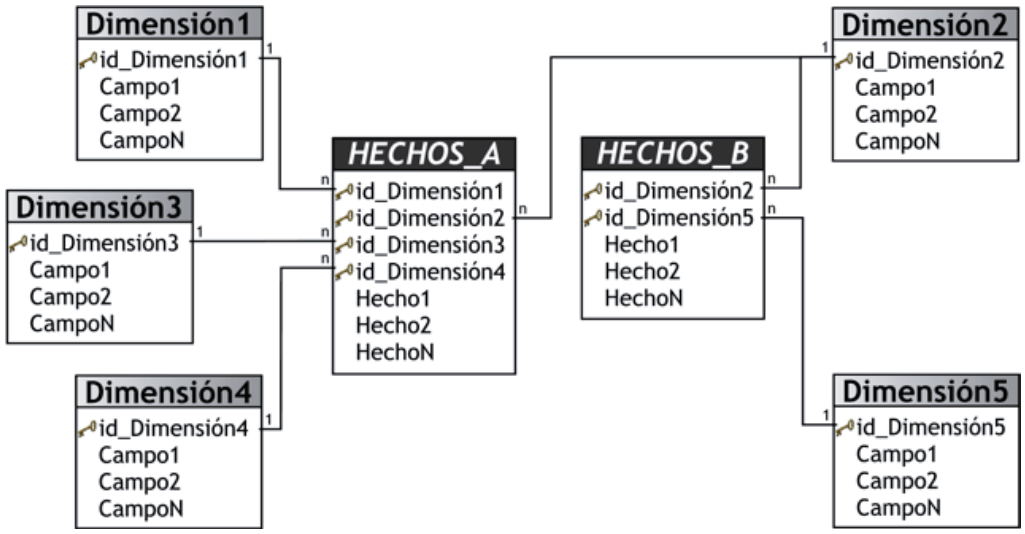


Figura 9 - Esquema de Constelación. Fuente: Elaboración propia.

Su diseño y cualidades son muy similares a las del esquema en estrella, pero posee una serie de diferencias con el mismo, que son precisamente las que lo destacan y caracterizan. 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 las tablas de dimensiones, ya que una misma tabla de dimensión puede utilizarse para varias tablas de hechos.
- ✓ No es soportado por todas las herramientas de consulta y análisis.

2.5.9 **OLTP v/s DWS**

Debido a que, ya se explicado y caracterizado los distintos tipos de esquemas del DW, se procederá a exponer las razones de su utilización, como así también las causas de por qué no se emplean simplemente las estructuras de las bases de datos tradicionales:

- ✓ Los OLTP son diseñados para soportar el procesamiento de información diaria de las empresas, y el énfasis recae en maximizar la capacidad transaccional de sus datos. Su estructura es altamente normalizada, para brindar mayor eficiencia a sistemas con muchas transacciones que acceden a un pequeño número de registros y están fuertemente condicionadas por los procesos operacionales que deben soportar, para la óptima actualización de sus datos. Esta estructura es ideal para llevar a cabo el proceso transaccional diario, brindar consultas sobre los datos cargados y tomar decisiones diarias, en cambio los esquemas de DW están diseñados para poder llevar a cabo procesos de consulta y análisis para luego tomar decisiones estratégicas y tácticas de alto nivel.

A continuación se presenta un cuadro resumen

	OLTP	Data Warehouse
Objetivo	Soportar actividades transaccionales diarias.	Consultar y analizar información estratégica y táctica.
Tipo de datos	Operacionales.	Para la toma de decisiones.
Modelo de datos	Normalizado.	Desnormalizado.
Consulta	SQL.	SQL más extensiones.
Datos consultados	Actuales.	Actuales e históricos.
Horizonte de tiempo	60 - 90 días.	5 - 10 años.
Tipos de consultas	Repetitivas, predefinidas	No previsibles, dinámicas
Nivel de almacenamiento	Nivel de detalle.	Nivel de detalle y diferentes niveles de sumarización.
Acciones disponibles	Alta, baja, modificación y consulta.	Carga y consulta.
Número de transacciones	Elevado	Medio o bajo
Tamaño	Pequeño - Mediano.	Grande.
Tiempo de respuesta	Pequeño (segundos - minutos).	Variable (minutos - horas).
Orientación	Orientado a las aplicaciones.	Orientado al negocio.
Sello de tiempo	La clave puede o no tener un elemento de tiempo.	La clave tiene un elemento de tiempo.
Estructura	Generalmente estable.	Generalmente varía de acuerdo a su propia evolución y utilización.

Figura 10 - OLTP v/s Data Warehouse. Fuente: Elaboración propia.

2.6 Consultas y Reportes

Corresponde a un análisis dirigido por el analista, y requiere tanto un conocimiento acabado de los datos como un trabajo excesivo sobre estos por parte de dicho analista.

Este análisis considera la definición de las consultas, el acceso y recuperación de datos, la manipulación de cálculos, y la preparación y entrega de los reportes. [3]

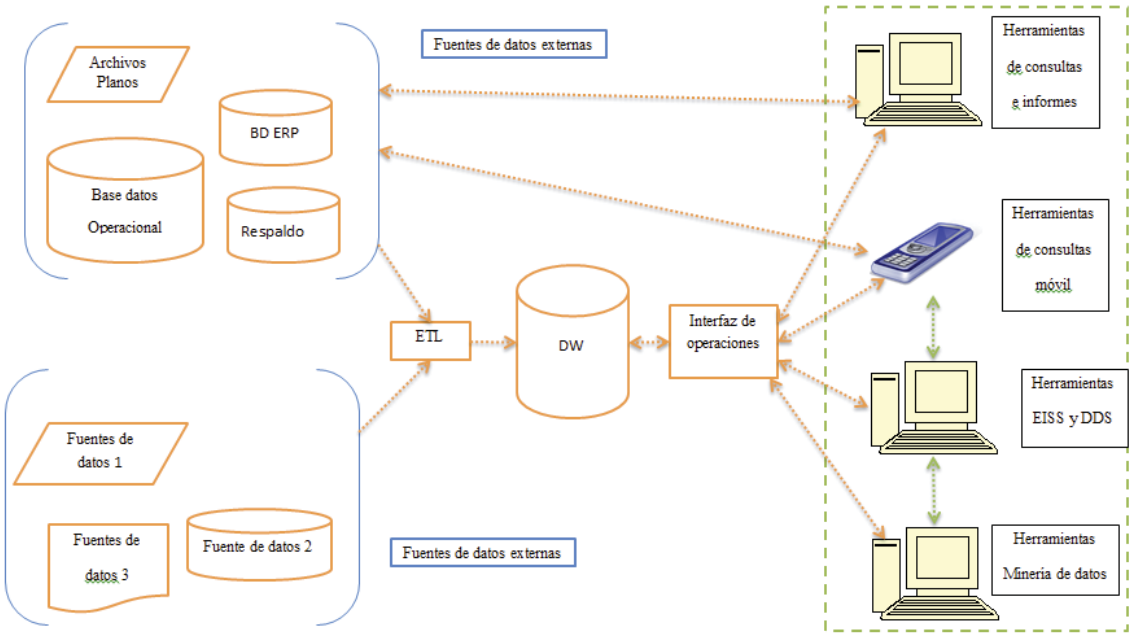


Figura 11 - Proceso Consulta y Reportes. Fuente: Elaboración propia.

2.7 Microsoft como herramienta de Inteligencia de Negocios

Microsoft Corporation es una empresa multinacional de origen estadounidense, fundada el 4 de abril de 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, tiene su sede en Redmond, Washington, Estados Unidos. Microsoft desarrolla, fabrica, licencia y produce software y equipos electrónicos, siendo sus productos más usados el sistema operativo Microsoft Windows y la suite Microsoft Office, los cuales tienen una importante posición entre los ordenadores personales. Con una cuota de mercado cercana al 90% para Office en 2003 y para Windows en 2006, siguiendo la estrategia de Bill Gates de *"tener una estación de trabajo que funcione con nuestro software en cada escritorio y en cada hogar"*.

2.8 SQL 2008 como herramienta de Inteligencia de Negocio.

SQL Server 2008 es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI). El motor de la base de datos SQL Server 2008 ofrece almacenamiento más seguro y confiable tanto para datos relacionales como estructurados, lo que le permite crear y administrar aplicaciones de datos altamente disponibles y con mayor rendimiento para utilizar en su negocio.

El motor de datos SQL Server 2008 constituye el núcleo de esta solución de administración de datos empresariales. Asimismo, SQL Server 2008 combina lo mejor en análisis, información, integración y notificación. Esto permite que su negocio cree y despliegue soluciones de BI rentables que ayuden a su equipo a incorporar datos en cada rincón del negocio a través de tableros de comando, escritorios digitales, servicios Web y dispositivos móviles.

La integración directa con Microsoft Visual Studio, el Microsoft Office System y un conjunto de nuevas herramientas de desarrollo, incluido el Business IntelligenceDevelopment Studio, distingue al SQL Server 2008. Ya sea que usted se desempeñe como encargado de desarrollo, administrador de base de datos, trabajador de la industria de la información o dirija una empresa, SQL Server 2008 ofrece soluciones innovadoras que le ayudan a obtener más valor de sus datos.

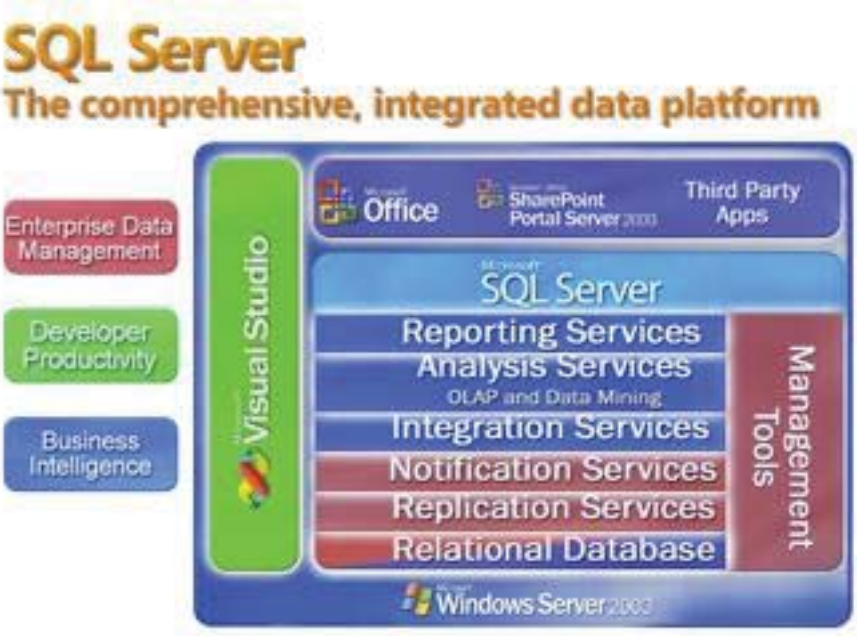


Figura 12 - Componentes básicos de SQL Server 2008.

2.9 Ediciones disponibles SQL Server 2008

Las ediciones disponibles de SQL Server 2008 están descriptas en el siguiente cuadro:

Edicion SQL Server 2008	Descripción
Enterprise Edition (Disponible en versiones 32-bit y 64-bit)	Una versión comprensiva de SQL Server diseñada para altos niveles de performance. Esta edición se usa para grandes escalas, niveles empresarios, aplicaciones críticas. La Enterprise Edition contiene todos las ventajas de la edición Standard, como así también las de Enterprise, incluyendo: • Failoverclustering • Databasemirroring • Snapshotdatabases • Mirroredbackups • Online page and file restore • Distributedpartitionedviews • Heterogeneousreplication • Peer-to-peer replication
Standard Edition (Disponible en versiones 32-bit y 64-bit versions)	Diseñado para aplicaciones de trabajo de grupo y departamentales. Use esta edición si no necesita los niveles de performance y disponibilidad ofrecidos por la Enterprise Edition.

Express Edition (Disponible en 32-bit)	Una versión de SQL Server 2008 para clientes que no están conectados o aplicaciones que se ejecutan solas.
Mobile Edition	Una edición compacta que provee administración de datos para equipos inteligentes. Esta edición es capaz de replicar datos con SQL Server 2008 y SQL Server 2000, permitiendo a los usuarios mantener un mobile data store que puede ser sincronizado con Enterprise data.
Developer Edition (Available in 32-bit and 64-bit versions)	Incluye todas las funcionalidades de la Enterprise Edition, pero esta licenciado para ser usado como un sistema de testeo y desarrollo, no como un servidor de producción. Use esta edición para desarrollar y testear soluciones para base de datos.

Tabla 1 - Ediciones disponibles en SQL Server 2008

2.10 Requerimiento de Hardware

Componente	Requerimiento
Procesador	Mínimo: 1.4 GHz (x64)
Memoria	Mínimo: 512 MB RAM. Máximo: 8 GB (Foundation) o 32 GB (Standard) o 2 TB (Enterprise, Datacenter, y Itanium-Based Systems)
Espacio en Disco	Mínimo: 32 GB o superior

Tabla 2 – Requerimiento mínimo de Hardware.

2.11 Inteligencia de Negocios en proyectos móviles

Los dispositivos móviles con el tiempo han ido ganando importancia, dado que la tecnología actual en los Smartphone permiten ver páginas, recibir y enviar correo, participar en redes sociales, ver películas y jugar en línea, las empresa por su lado han visto el potencial y le dan dado el uso para la comunicación empresarial y uno de estos es el uso de la inteligencia de negocio.

Utilizar el iPad puede parecer un capricho de moda, pero por sus características técnicas, resulta muy útil para adoptar una aplicación usando inteligencia de negocio, por las dimensiones de su pantalla, calidad de imagen, ligero peso, duración de batería y sobre todo, por la forma en que se interactúa con las aplicaciones.

La libertad de contar con información online y vía WiFi es un diferencial importante. Los ejecutivos tienen acceso desde su escritorio, las salas de reunión o los pasillos, a todos los datos disponibles.

La distancia entre preguntas y respuestas se acorta, se mejora la retroalimentación de sus procesos de pensamiento, optimizando el análisis y la toma de decisiones. A eso hay que sumarle la facilidad y versatilidad de uso de las tablets, que se ve maximizada en la plataforma Apple.

No se trata en asegurar que el BI de escritorio o tradicional desaparecerá, al contrario, el BI móvil lo refuerza. Creer que en el mediano plazo se contemplarán, como una fase propia de un proyecto de inteligencia de negocios, el despliegue en dispositivos móviles.

El BI con el iPad, ha dado grandes pasos, pero aún estamos en el umbral de esta tecnología. Los dispositivos móviles con el tiempo han ido ganando importancia, dado que la tecnología actual en los Smartphone permiten ver páginas, recibir y enviar correo, participar en redes sociales, ver películas y jugar en línea, las empresa por su lado han visto el potencial y le dan dado el uso para la comunicación empresarial y uno de estos es el uso de la inteligencia de negocio.

Figura 13 - Proyecto Inteligencia de Negocios en iPhone.

CAPÍTULO 3

3 Herramientas de desarrollo

3.1 Xcode

El paquete de herramientas Xcode proporciona todo lo necesario para el desarrollo de creación de aplicaciones Mac, iPhone e iPad.

Xcode está estrechamente integrado con Cocoa and Cocoa touch Framework, creación de un entorno productivo y fácil de usar que es lo suficientemente potente como pasa ser las mismas herramientas utilizadas por Apple para producir OS X e IOS. El conjunto de herramientas Xcode incluye el IDE, herramienta de diseño con un diseño de construcción de interfaces y completamente integrado un Apple LLVM. La herramienta de análisis de instrumentos también se incluye, junto con docenas de otras herramientas de desarrollo de apoyo.

Diseñado desde cero para aprovechar las nuevas tecnologías de Apple, Xcode integra todas las herramientas que necesita. La interfaz unificada sin problemas las transiciones de componer el código fuente, para la depuración, e incluso el diseño de su interfaz de usuario impresionante, todo dentro de la misma ventana [4].

Xcode al igual que otros programas cuenta con exploración de variables en entorno de ejecución o modo debug, aviso de errores en la codificación entre otros, permitiendo al usuario saber lo que hace y publicar las aplicaciones en la tienda de Mac (Apple Store).

La nueva tecnología en compiladores de Apple se integra en la experiencia del desarrollo. El analizador que utiliza para construir C/C++ y Objective-C posee un motor de indexación proporcionando terminaciones de código increíblemente precisas. A medida que se trabaja en el desarrollo la Apple LLVM está constantemente evaluando lo que se escribe, identificando errores de codificación en tiempo real y pensando en futuras formas de Fix-it para el desarrollador, mientras que otros compiladores dicen lo que está mal, Apple LLVM puede hacer las cosas bien.

Xcode posee análisis de desempeño y comportamiento permitiendo localizar cuellos de botella de rendimiento de OS X y las aplicaciones iOS.

La Herramienta Xcode posee un simulador donde se puede correr las aplicaciones creadas dependiendo el dispositivo escogido ya un iPhone o un iPad mejorando la experiencia de usuario.

Figura 14 - Herramienta Xcode.

3.2 Objective C

3.2.1 **Historia**

Objective-C es un lenguaje de programación orientado a objetos creado como un superconjunto de C para que se implemente un modelo de objeto parecido al de Smaltalk. Originalmente fue creado por Bard Cox y por la corporación StepStone en 1980. En 1988 fue adoptado como lenguaje de programación de NEXTSTEP y en 1992 fue liberado bajo licencia GPL para el compilador GCC. Actualmente se usa como lenguaje principal de programación en Mac OSX, iOS y GNUstep.

La programación estructurada se estableció para ayudar a dividir los programas en pequeñas partes, haciendo más fácil el desarrollo cuando la aplicación se volvía muy grande. Sin embargo, como los problemas seguían creciendo al pasar el tiempo, la programación estructurada se volvió compleja dado el desorden de algunos programadores para invocar instrucciones repetitivamente, llevando a código spaghetti y dificultando la reutilización de código.

Muchos vieron que la programación orientada a objetos sería la solución al problema. De hecho, Smalltalk ya tenía solucionados muchos de estos problemas: algunos de los sistemas más complejos en el mundo funcionaban gracias a Smalltalk. Pero Smalltalk usaba una máquina virtual, lo cual requería mucha memoria para esa época, y era demasiado lento.

Objective-C fue creado principalmente por Brad Cox y Tom Love a inicios de los 80 en su compañía Stepstone. Ambos fueron iniciados en Smalltalk mientras estaban en el Programming Technology Center de ITT en 1981. Cox se vio interesado en los problemas de reutilización en el desarrollo de software. Se dio cuenta de que un lenguaje como Smalltalk sería imprescindible en la construcción de entornos de desarrollo potentes para los desarrolladores en ITI Corporation. Cox empezó a modificar el compilador de C para agregar algunas de las capacidades de Smalltalk. Pronto tuvo una extensión para añadir la programación orientada a objetos a C la cual llamó «OOPC» (*Object-Oriented Programming in C*). Love mientras tanto, fue contratado por Shlumberger Research en 1982 y tuvo la oportunidad de adquirir la primera copia de Smalltalk-80, lo que influyó en su estilo como programador. [4]

Para demostrar que se hizo un progreso real, Cox mostró que para hacer componentes de software verdaderamente intercambiables sólo se necesitaban unos pequeños cambios en las herramientas existentes.

Específicamente, estas necesitaban soportar objetos de manera flexible, venir con un conjunto de bibliotecas que fueran utilizables, y permitir que el código (y cualquier recurso necesitado por el código) pudiera ser empaquetado en un formato multiplataforma.

Cox y Love luego fundaron una nueva empresa, Productivity Products International (PPI), para comercializar su producto, el cual era un compilador de Objective-C con un conjunto de bibliotecas potentes.

En 1986, Cox publicó la principal descripción de Objective-C en su forma original en el libro *Object-Oriented Programming, An Evolutionary Approach*. Aunque él fue cuidadoso en resaltar que hay muchos problemas de reutilización que no dependen del lenguaje, Objective-C frecuentemente fue comparado detalladamente con otros lenguajes.

3.2.2 **Sintaxis**

Objective-C consiste en una capa situada por encima de C, y además es un estricto superconjunto de C. Esto es, es posible compilar cualquier programa escrito en C con un compilador de Objective-C, y también puede incluir libremente código en C dentro de una clase de Objective-C.

Por ejemplo el siguiente código:

```
#import <stdio.h>
int main( int argc, const char *argv[] ) {
   printf( "Hola Mundo\n" );
   return 0;
}
```

El código anterior se diferencia de un código en C común por la primera instrucción #import, que difiere del #include del C clásico, pero la función printf("") es puramente C. La función propia de Objective-C para imprimir una cadena de caracteres en consola es NSLog(@""); utilizándola, el código anterior quedaría de la siguiente manera:

```
int main( int argc, const char *argv[] )
{
   NSLog( @"Hola Mundo\n" );
   return 0;
}
```

La sintaxis de objetos de Objective-C deriva de Smalltalk. Toda la sintaxis para las operaciones no orientadas a objetos (incluyendo variables primitivas, pre-procesamiento, expresiones, declaración de funciones y llamadas a funciones) son idénticas a las de C, mientras que la sintaxis para las características orientadas a objetos es una implementación similar a la mensajería de Smalltalk. [4]

3.3 Java

3.3.1 **Introducción**

La tecnología Java se creó como una herramienta de programación en una pequeña operación secreta y anónima denominada "the Green Project" en Sun Microsystems en el año 1991.
El equipo secreto ("Green Team"), compuesto por trece personas y dirigido por James Gosling, se encerró en una oficina desconocida de Sand Hill Road en Menlo Park, interrumpió todas las comunicaciones regulares con Sun y trabajó sin descanso durante 18 meses.

Intentaban anticiparse y prepararse para el futuro de la informática. Su conclusión inicial fue que al menos en parte se tendería hacia la convergencia de los dispositivos digitales y los computadores. El resultado fue un lenguaje de programación que no dependía de los dispositivos denominado "Oak".

Para demostrar cómo podía contribuir este nuevo lenguaje al futuro de los dispositivos digitales, el equipo desarrolló un controlador de dispositivos de mano para uso doméstico destinado al sector de la televisión digital por cable. Por desgracia, la idea resultó ser demasiado avanzada para el momento y el sector de la televisión digital por cable no estaba listo para el gran avance que la tecnología Java les ofrecía.

Pero poco tiempo después Internet estaba listo para la tecnología Java y, justo a tiempo para su presentación en público en 1995, el equipo pudo anunciar que el navegador Netscape Navigator incorporaría la tecnología Java.

Actualmente, a punto de cumplir los 10 años de existencia, la plataforma Java ha atraído a cerca de 4 millones de desarrolladores de software, se utiliza en los principales sectores de la industria de todo el mundo y está presente en un gran número de dispositivos, computadores y redes de cualquier tecnología de programación.

De hecho, su versatilidad y eficiencia, la portabilidad de su plataforma y la seguridad que aporta, la han convertido en la tecnología ideal para aplicación en redes, de manera que hoy en día, más de 2.500 millones de dispositivos utilizan la tecnología Java.

- ✓ Más de 700 millones de computadores.
- ✓ 708 millones de teléfonos móviles y otros dispositivos de mano (fuente: Ovum).
- ✓ 1000 millones de tarjetas inteligentes.
- ✓ Además de sintonizadores, impresoras, webcams, juegos, sistemas de navegación para automóviles, terminales de lotería, dispositivos médicos, cajeros de pago en estacionamientos, etc.

Hoy en día, puede encontrar la tecnología Java en redes y dispositivos que comprenden desde Internet y supercomputadores científicos hasta notebooks y teléfonos móviles; desde simuladores de mercado en Wall Street hasta juegos de uso doméstico y tarjetas de crédito: Java está en todas partes.
El mejor modo de conocer todas estas aplicaciones es accediendo a java.com, un lugar fundamental para realizar compras y conocer los productos que, asimismo, constituye un recurso de información central para empresas, consumidores y desarrolladores de software que utilicen la tecnología Java.

¿**Por qué los desarrolladores de software eligen la tecnología Java?**

El lenguaje de programación Java ha sido totalmente mejorado, ampliado y probado por una comunidad activa de unos cuatro millones de desarrolladores de software.

La tecnología Java, una tecnología madura, extremadamente eficaz y sorprendentemente versátil, se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- ✓ Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- ✓ Crear programas para que funcionen en un navegador Web y en servicios Web.
- ✓ Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, etc.
- ✓ Combinar aplicaciones o servicios basados en la tecnología Java para crear servicios o aplicaciones totalmente personalizados.
- ✓ Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo costo y prácticamente cualquier dispositivo digital.

3.3.2 **Introducción a los Servlet**

Entre las muchas particularidades que distinguen a la Web de los restantes medios de comunicación, está la capacidad de interacción. Interacción que inicialmente era bastante reducida y se conseguía a través de código HTML y algunos elementos embebidos de JavaScript. Cuando se mencionaba el nombre de Java en este entorno, algunos lo asociaban a JavaScript, y otros, un poco más conocedores, lo asociaban con pequeños programas que se ejecutaban en el cliente y se llamaban Applets. Esto realmente fue el inicio de Java en la Web y surgió para darle más vida y dinamismo a la misma. [5]

Esto resolvería casi todos los problemas al comienzo de la vida en la Web, y hacia el trabajo con Java bastante limitado pero, ¿realmente una página tiene que limitarse sólo a contener textos y gráficos bonitos? Esta necesidad se comprendió muy pronto gracias al dinamismo de la informática y las exigencias de los usuarios, y comenzaron a aparecer tecnologías que convertían al usuario en la pieza principal de la aplicación, tecnologías que podían interpretar sus acciones y procesas datos no sólo en el cliente sino también en el servidor. Los CGI de Perl eran la solución en ese entonces. La comunidad de Java no podía quedarse atrás y se lanza la especificación original de Sevlet en junio de 1997, gracias al ingenio de los programadores de la SUN, en su versión 1.0.

Los Servlet son la respuesta de la tecnología en Java a la programación de la Interfaz de Compuerta Común (CGI). Son programas que se ejecutan en el servidor, realizando la función de una capa intermedia entre una petición proveniente de un navegador Web u otro cliente HTTP, y las aplicaciones del servidor, pudiendo utilizar toda la paquetería y potencialidades del lenguaje. Su función principal es proveer páginas web dinámicas y personalizadas, utilizando para este objetivo el acceso a bases de datos, flujos de trabajo y otros recursos [5].

3.3.3 **Metodología de desarrollo de un Servlet**

El API Servlet nos propone una jerarquía de clases bien definidas para el trabajo con los mismos, donde se puede encontrar desde Servlets sin implementaciones definidas, en los cuales se tiene que realizar todo el trabajo como la clase GenericServlet, o clases un poco más avanzadas como la HttpServlet, donde sólo tendrá que implementar los métodos de acceso al mismo como el doGet() o el doPost(). El principal componente del API es la interfaz Servlet. La misma esta provista de los principales métodos para manipular, no sólo los Servlets, sino también la comunicación de estos con los clientes. Todos los Servlets implementan esta interfaz de una manera directa o indirecta.

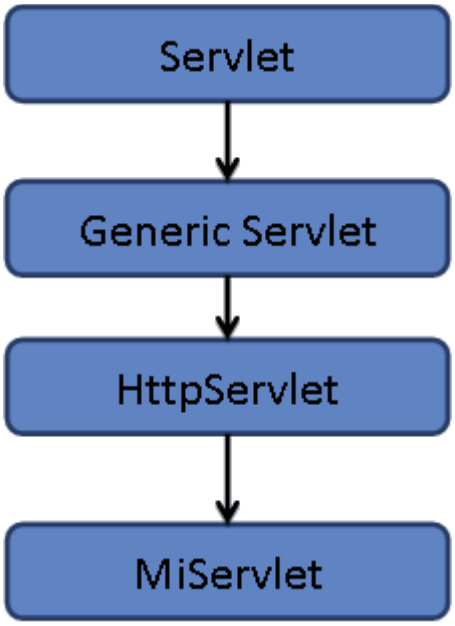


Figura 15 – Jerarquía. Fuente: Elaboración propia.

3.3.4 **Característica de los Servlet**

- **Ventajas sobre los CGI tradicionales:**

En el momento que salió la especificación de Servelt a la luz, los CGI ya llevaban algún tiempo en la red, lo que dio una ventaja muy grande pues permitió que fueran analizados a profundidad y de esta forma poder crear métodos con mejor rendimiento, potencialidad y facilidad de uso.

- **Eficientes:**

Con el CGI tradicional, se arranca un nuevo proceso por cada petición HTTP. Si el propio programa CGI es relativamente pequeño, el costo de arrancar el proceso puede superar el tiempo de ejecución. Además, si ocurre n peticiones simultáneas al mismo programa este carga su código en memoria la misma cantidad de veces. Con los Servlet la máquina virtual de java permanece en ejecución y administra cada petición mediante ligeros subprocesos de Java y no un pesado proceso del sistema operativo. A través de la programación por hilos pueden existir n subprocesos del mismo Servlet y este encontrarse en memoria una sola vez. Esto nos da una velocidad de respuesta mayor a cada petición y una eficiencia superior en el aprovechamiento de los recursos de la maquina donde son ejecutados.

- **Adecuados:**

Leer los datos enviados por un cliente HTTP desde un CGI tradicional es un trabajo bastante engorros, se necesita parsear la cadena completa y extraer de ella los datos uno a uno. Los Servlet cuentan con una extensa infraestructura para analizar y decodificar automáticamente los datos provenientes de un formulario HTML, leer y establecer encabezados HTTP, administrar las cookies, rastrear las sesiones y muchas otras utilidades de alto nivel como estas.

- **Transportables:**

Los Servlet no son más que una clase Java, por lo que pueden ser tan portables como cualquier aplicación escrita en el mismo lenguaje. Esto los convierte en multiplataforma por defecto por lo que pueden ser ejecutados en cualquier sistema operativo. Además cuentan con un API estándar. La API Servlet no incluye nada acerca de cómo son cargados los Servlets, ni el ambiente en el cual corren los Servlets, ni el protocolo usado para transmitir los datos del usuario. Esto permite a los Servlets poder ser usados por diferentes servidores Web. Se pueden cargar indiferentemente y de forma transparente tanto desde un disco local como desde una dirección remota, de forma totalmente transparentes. Es posible utilizarlos en servidores tan populares como Apache, FastTrack o Internet Information Server (IIS).

Los Servlets pueden comunicarse entre sí, y por tanto, es posible una reasignación dinámica de la carga de proceso entre diversas máquinas. Es decir, un Servlet podría pasarle trabajo a otro Servlet residente en otra máquina conociendo solamente la URL de este.

- **Seguros**:

Una de las principales fuentes de vulnerabilidad en los programas CGI tradicionales proviene del hecho de que por lo general son ejecutados por entornos de sistemas operativos de propósito general. Por ello, el programador de CGI debe tener cuidado de filtrar caracteres como las comillas tipográficas y los puntos y comas pues tienen un tratamiento especial por el entorno. Esto es más complejo de lo que podría pensarse, y los problemas derivados de esta situación son constantemente ignorados en diversas bibliotecas de CGI. Una segunda fuente de problemas es el hecho de que ciertos programas de CGI son procesados por lenguajes que no verifican automáticamente los límites de las matrices o cadenas. Por ejemplo, en C y C++ es perfectamente legal asignar una matriz de 100 elementos y describir en el "elemento" número 999, el cual puede ser un lugar aleatorio de la memoria del programa. Por ello, los programadores que olvidan realizar esta verificación, abren sus propios sistemas a posibles ataques de desbordamiento en el búfer ya sea por accidente o de manera deliberada. Los Servlets no tienen estos problemas. Aun si un Servlet ejecuta una llamada a un sistema distante para ejecutar un programa en el sistema operativo local, no utiliza el entorno del sistema operativo para lograrlo. Y por supuesto que la verificación de los límites de las matrices y otras características para la protección de la memoria es una parte central del lenguaje de programación Java.

3.3.5 Equivalencia entre CGI y Servlet

Si ya ha desarrollado varios trabajos donde ha utilizado CGI y no desea que sean desechados, con los Servlet puede comunicarse e incluso incrustarlos en el código de un Servlet.

CGI	Servlet	Significado
AUTH_TYPE	request.getAuthType()	Si se suministró una cabecera **Authorization**, este es el esquema especificado (**basic** o **digest**)
CONTENT_LENGTH	request.getContentLength()	Sólo para peticiones **POST**, el número de bytes enviados.
CONTENT_TYPE	request.getContentType()	El tipo MIME de los datos adjuntos, si se específica.
DOCUMENT_ROOT	getServletContext().getRealPath("/")	Camino al directorio que corresponde con **http://host/**
PATH_INFO	request.getPathInfo()	Información del camino adjunto a la URL. Como los Servlets, al contrario que los programas estándares CGI, pueden hablar con el servidor, no necesitan tratar esto de

		forma separada. La información del path podría ser enviada como parte normal de los datos de formulario
REMOTE_ADDR	request.getRemoteAddr()	La dirección IP del cliente que hizo la petición, por ejemplo "**192.9.48.9**".
REMOTE_HOST	request.getRemoteHost()	El nombre de dominio totalmente cualificado (por ejemplo "**java.sun.com**") del cliente que hizo la petición. Se devuelve la dirección IP si no se puede determinar.
REMOTE_USER	request.getRemoteUser()	Si se suministró una cabecera **Authorization**, la parte del usuario.
REQUEST_METHOD	request.getMethod()	El tipo de petición, **GET**, **POST**, **HEAD**, **PUT**, **DELETE**, **OPTIONS**, o **TRACE**
SERVER_NAME	request.getServerName()	Nombre del Servidor Web.
SERVER_PORT	request.getServerPort()	Puerto por el que escucha el servidor.
SERVER_PROTOCOL	request.getProtocol()	Nombre y versión usada en la línea de petición (por ejemplo **HTTP/1.0** o **HTTP/1.1**).

Tabla 3 - Cuadro comparativo entre CGI y Servlet.

3.3.6 **Ciclo de vida de un servlet**

El proceso de ejecución de un Servlet comienza con la petición HTTP. Este llega en primer lugar al servidor web, el cual la traslada al motor del servicio Servlet/JSP. El motor encapsula la petición en un objeto del tipo HttpServletRequest y el flujo de respuesta en un objeto HttpServletRespopnse.

.La interfaz ServletRequest permite al Servlet acceder a la información pasada por el cliente como, los nombres de parámetros, el protocolo, los nombres de los host remotos que hacen la solicitud y el servidor que la recibe. Esta interfaz permite a los Servlets el acceso a métodos que permiten manejar la presentación de la respuesta como salida en el navegador, a través de los cuales consiguen los datos desde el cliente que usa protocolos como HTTP POST [5].

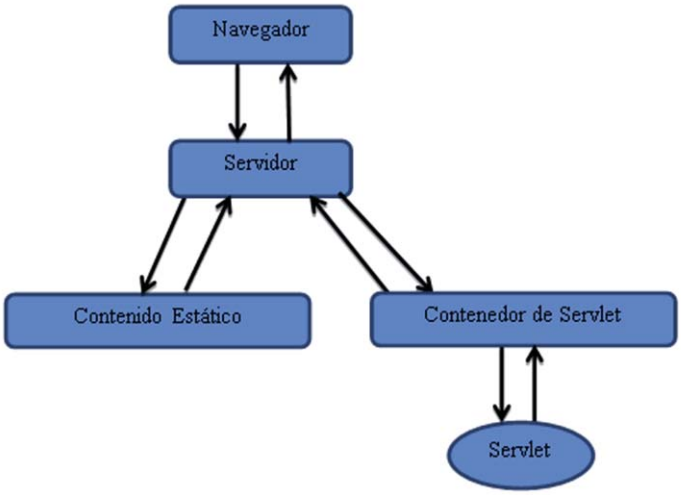


Figura 16 - Peticiones 1. Fuente: Elaboración propia.

La interfaz ServletResponse proporciona al Servlet los métodos para responderle al cliente. Permite configurar el formato de salida de los datos. ServletOutputStream permite enviar la réplica de datos como respuesta. Las subclases de ServletResponse le dan más capacidad al Servlet para responder.

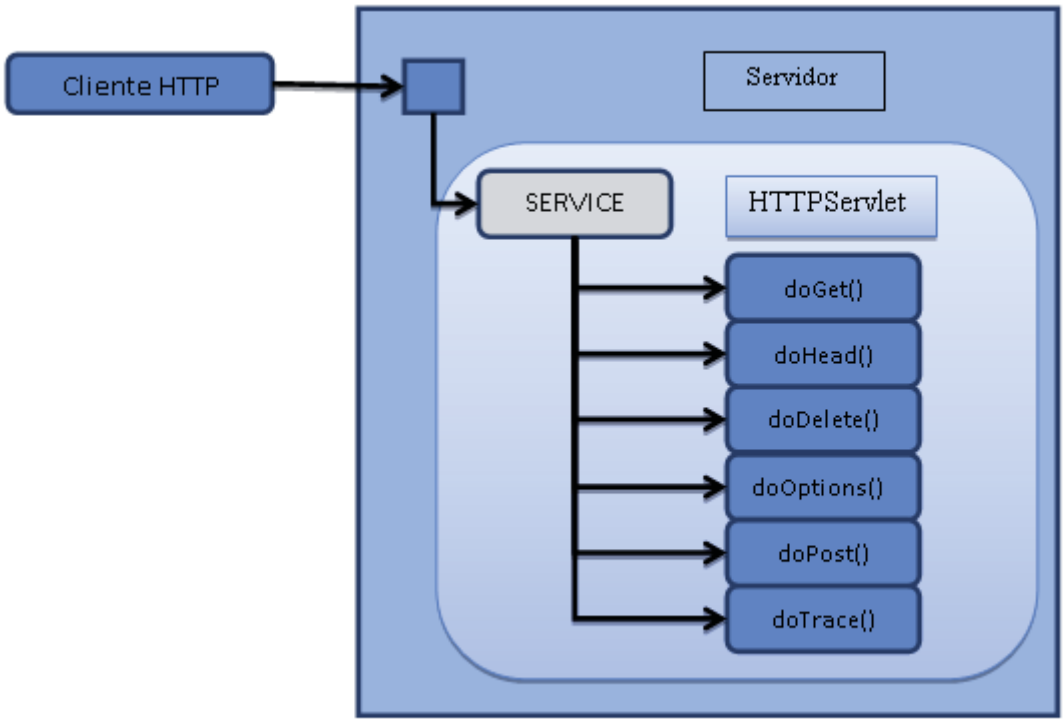


Figura 17 - Peticiones 2. Fuente: Elaboración propia.

El Servlet es creado la primera vez que se invoca. Esto se hace a través del método **init()** que es el encargado de cargarlo en la memoria y pasarle los parámetros iniciales que se le hallan definido. Luego de que el Servlet es inicializado, todas las demás peticiones serán tratadas en subprocesos o hilos que solo necesitaran como entrada el método **service()**. Este método invoca al doGet(), al doPost() o al método definido por el cliente para manejar la petición entrante.

Las peticiones serán manejadas como se ha explicado hasta ahora, pero ¿qué se hace con un Servlet que no necesitamos que se ejecute más? ¿Lo tendríamos ocupando espacio en la memoria? Para resolver esta problemática existe el método **destroy()**. Este método es el encargado de destruir el Servlet. Este es el momento indicado para cerrar las conexiones que existan con bases de datos u otros recursos compartidos, salvar el estado o lo que queramos que persista.

Las clases e interfaces descritas conforman a un Servlet básico. Pero existen métodos adicionales que provee la API con la capacidad para controlar sesiones o múltiples conexiones, entre muchas más aplicaciones.

3.3.7 **Obteniendo datos del cliente**

Una de las formas más comunes, que encontramos en la web, de sitios dinámicos es a través de la interacción con formularios. Los Servlet cuentan con una agradable comunicación con estos parámetros pasados por el usuario. Simplemente se necesita llamar al método getParameter del HttpServletRequest y proporcionarle el nombre del parámetro que se desea obtener como argumento de la función. Este método se utilizara tanto para peticiones GET como POST, los Servlet son capaces de reconocer como tienen que realizar la obtención de los mismos.

Por supuesto que no es el único método que existe para la obtención de parámetros pasados por el cliente

- **getParameter:** Devuelve el valor del parámetro que fue pasado por argumento
- **getParameterValues:** Devuelve todos los valores que tiene un parámetro específico
- **getParameterNames:** Devuelve los nombres de todos los parámetros que fueron enviados por el usuario.

3.3.8 **El trabajo con las cabeceras**

Una de las claves para realizar aplicaciones Web efectivas es comprender como manejar el Protocolo para la Transferencia de Hipertexto (HTTP). Tener un dominio de este protocolo no se convierte, en el caso de los Servlet, en algo abstracto, sino en prácticas que pueden tener un impacto inmediato en el rendimiento y funcionalidad de las aplicaciones.

Dentro de un mensaje HTTP los encabezados son muy importantes. Definen en gran parte la información que se intercambia entre clientes y servidores dándole flexibilidad al protocolo. Estas líneas permiten que se envíe información descriptiva en la propia transacción, permitiendo cosas como la autenticación, identificación de usuarios, tipos de MIME soportadas, si la conexión es persistente o no, etc. La sintaxis de una línea simple de encabezado se describe de la siguiente manera:

Nombre-de-Encabezado: Valor.

Cabe destacar que los datos de encabezados de petición HTTP son diferentes a los pasados por el cliente como parámetros. Estos últimos se envían al servidor como parte de la URL en el caso de las peticiones GET, y en una línea por separado en peticiones POST. Los encabezados de petición, por otro lado, son establecidos indirectamente por el navegador Web y se envían inmediatamente después de la línea de petición inicial (GET, POST, etc.).

La lectura de los encabezados es un proceso directo; tan sólo ejecute al método getHeader() de HttpServletRequest, que devolverá una cadena si el encabezado indicado se ha proporcionado en esta petición o , de lo contrario devolverá null. Aunque getHeadres() es la forma genérica de leer los encabezados de entrada, existen algunos que son utilizados con más frecuencia y se le han desarrollado algunos métodos en HttpServletRequest para el acceso rápido a los mismos.

- **getCookies:**

  Devuelve el contenido del encabezado Cookie, analizado y almacenado en una matriz de objetos Cookie.

- **getAuthType y getRemoteUser:**

  Dividen el encabezado Authorization en sus secciones constitutivas.

- **getContentLength**:

  Devuelve el valor del encabezado Content-Length como un valor int.

- **getContentType:**

  Devuelve el valor del encabezado Contet-Type como una cadena de caracteres.

- **getDateHeader y getIntHeader**:

  Leen el encabezado especifico y luego lo convierten a valores Date e int respectivamente.

- **getHeaderNames:**

  Devuelve una enumeración de todos los nombres de los encabezados recibidos en una petición en particular.

- **getHeaders:**

  Cada nombre de encabezado aparece una sola vez por petición, aunque ocasionalmente pueden aparecer varios valores para un encabezado en particular. Accept-Languaje es un buen ejemplo de esto. Para resolver este problema, desde la versión 2.2 de la especificación de Servlet, este método devuelve enumerados todos los valores de todos los encabezados encontrados con el mismo nombre. Además de buscar los encabezados de petición, se puede obtener información de la línea principal de la petición, también por medio de los métodos de HttpServletRequest.

- **getMethod:**

  Devuelve el método de petición principal (GET, POST, etc.).

- **getRequestURI:**

  Devuelve la parte de la URL después del host y el puerto pero antes de los datos del formulario.

- **getProtocol:**

  Devuelve el protocolo utilizado (HTTP/1.0, HTTP/1.1).

3.3.9 **Códigos de Estados**

Cuando un servidor Web responde a una petición, la respuesta, por lo general, consta de una línea de estado, algunos encabezados de respuesta y el documento.

HTTP/1.1 200 OK
Content-Type: text/plain
Hola Mundo!!!

Los Servlets pueden ejecutar varias tareas de importancia mediante el manejo de la línea de estado. Por ejemplo, renviar al usuario a otro sitio, indicar que el documento no se encuentra disponible, que el usuario requiere contraseña, etc.

Como se mostró en el ejemplo anterior, la línea de estado de respuesta HTTP consta de una versión HTTP, un código de estado y un mensaje asociado. Dado que el mensaje está asociado directamente con el código de estado y la versión HTTP son definidos por el servidor; todo lo que un Servlet necesita hacer es definir este código. La forma de hacerlo es a través del método set Status de HttpServletRespopnse. Asegúrese de establecer los códigos de estado antes de enviar cualquier parte del documento al cliente.

Aunque el método general para establecer códigos de estado es de la forma response.setStatus(int), existen dos casos en los que se cuenta con métodos de acceso rápido por su común uso.

- **public void sendError(int code, String Message):**

Envía un código de estado (404 generalmente) junto a un mensaje que se formatea automáticamente dentro de un documento HTML.

- **public void sendRedirect(String url):**

El método genera una respuesta 302 junto a un encabezado Location que devuelve el URL del nuevo documento. En la versión actual del API Servlet esta URL puede ser absoluta o relativa.

Si desea realizar una redirección dentro de su propio sitio y además mantener un rastreo de Sesión, utilice como parámetro del senRedirect el siguiente código: response.encodeURL(url)

3.4 IOS

3.4.1 **Introducción**

IOS (anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple desarrollado originalmente para el iPhone, siendo después usado en todos los dispositivos iPhone, iPod Touch e iPad. Es un derivado de Mac OS X, que a su vez está basado en Darwin BSD. El iOS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios de comunicación" y la capa de "CocoaTouch". Todo el sistema se encuentra en la partición "/root" del dispositivo, ocupa poco menos de 500 megabytes. [1]

Características:

- Interfaz de usuario:

  La interfaz de usuario de iOS se basa en con el concepto de manipulación mediante gestos multitáctil. Los elementos de la interfaz se componen por deslizadores, interruptores y botones. La respuesta es inmediata y se provee de una interfaz fluida. La interacción con el sistema operativo se realiza mediante gestos como deslizar, tocar y pellizcar. Acelerómetros y Giroscopios internos son utilizados por algunas aplicaciones para responder a movimientos y gestos, como sacudir el aparato (en campos de texto es usado para deshacer y rehacer) o rotarlo (se suele usar para cambiar de posición vertical a modo paisaje).

- Pantalla principal:

  La pantalla principal (llamada «SpringBoard») es donde se ubican los iconos de las aplicaciones y el Dock en la parte inferior de la pantalla donde se pueden anclar aplicaciones de uso frecuente, aparece al desbloquear el dispositivo o presionar el botón de inicio. La pantalla tiene una barra de estado en la parte superior para mostrar datos, tales como la hora, el nivel de batería, y la intensidad de la señal.

- Aplicaciones:

Nombre	Función	Versión de introducción
Teléfono, FaceTime Videoconferencia	1.0+ (Videoconferencia: 4.0+)	
Safari	Navegador Web	1.0+
Mail	Cliente de correo electrónico	1.0+
IPod	Reproductor de medios	1.0+

Nombre	Función	Versión de introducción
Mensajes	Mensajes de texto	MMS 1.0+ (MMS 3.0+, iMessage 5.0+)
Calendario	Calendario	1.0+
Fotos	Visor de fotos	1.0+ (Visor de video 3.0+)
Cámara	Cámara	1.0+ (Grabación de video 3.0+, video en HD 4.0+)
YouTube	Visor de videos en YouTube	1.0+
Bolsa	Visor de cotizaciones en bolsa	1.0+
Mapas	Google Maps	1.0+ (GPS 2.0+, Brújula 3.0+)
Tiempo	Yahoo! Weather	1.0+
Notas de Voz	Grabador de voz	3.0+
Notas	Notas en texto plano	1.0+
Reloj	Reloj mundial, cronómetro, alarma y temporizador	1.0+
Calculadora	Calculadora (incluye versión científica)	1.0+
Ajustes	Ajustes del dispositivo	1.0+
iTunes	Para acceder al iTunes Music Store	1.1+
App Store	Comprar y/o descargar apps	2.0+
Brújula	Brújula	3.0+
Contactos	Lista de teléfonos, direcciones y contactos	1.0+
Nike iPod	Nike iPod	3.0+
iBooks	Visualizar PDF y obtener E-Books	4.0+
Game Center	Red social de juegos	4.1+
Facetime	Video llamadas sobre WI-FI	4.1+
Recordatorios	Aplicación de tipo To-Do list	5.0+
Quiosco	Suscripciones de diarios y revistas	5.0+

Tabla 4 - Aplicaciones básicas de un iPhone.

- Multitarea :

Antes del IOS 4 la multitarea estaba reservada para las aplicaciones por defecto del sistema. A partir del IOS 4, dispositivo de tercera generación y posteriores soportan el uso de 7 Apis para multitarea, específicamente:

  - Audio n segundo plano
  - Voz IP
  - Localización en segundo plano
  - Notificaciones Push
  - Notificaciones locales
  - Completado de tareas
  - Cambio rápido de aplicaciones

Las tareas que son ajenas del SO quedan congeladas en segundo plano no recibiendo un sólo ciclo del reloj del reloj.

- Game Center:

Es una red social basa en juegos. En iOS 5 se perfecciono, pudiendo agregar una foto a tu perfil, pudiendo ver los amigos de tus amigos y pudiendo encontrar adversarios con recomendaciones de nuevos amigos en función de tus juegos y jugadores favoritos.

3.4.2 **Smartphone**

Vamos a intentar dejar claro que es un Smartphone o también llamado teléfono inteligente (Smartphone en inglés) es un término comercial para denominar a un teléfono móvil que ofrece más funciones que un teléfono móvil común. Casi todos los teléfonos inteligentes son móviles que soportan completamente un cliente de correo electrónico con la funcionalidad completa de un organizador personal.

La característica más importante (una de ellas) de casi todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero.

El término "Inteligente" hace referencia a cualquier interfaz, como un teclado QWERTY en miniatura, una pantalla táctil (lo más habitual, denominándose en este caso "teléfono móvil táctil"), o simplemente el sistema operativo móvil que posee, diferenciando su uso mediante una exclusiva disposición de los menús, teclas, atajos, etc.

El completo soporte al correo electrónico parece ser una característica indispensable encontrada en todos los modelos existentes y anunciados en 2007, 2008, 2009 y 2010.

Casi todos los teléfonos inteligentes también permiten al usuario instalar programas adicionales, normalmente inclusive desde terceros, pero algunos vendedores gustan de tildar a sus teléfonos como inteligentes aun cuando no tienen esa característica.

Algunos ejemplos de teléfonos denominados inteligentes son: Serie MOTO Q de Motorola, Nokia series E y series N, BlackBerry, Samsung Wave, iPhone y todos los que tienen el sistema operativo Android, como por ejemplo: Google NexusOne, Motorola Milestone y Sony Ericsson XperiaArc.

Smartphone de un teléfono móvil:

- ✓ Soporta correo electrónico
- ✓ Cuenta con GPS
- ✓ Permiten la instalación de programas de terceros
- ✓ Utiliza cualquier interfaz para el ingreso de datos, como por ejemplo teclado QWERTY, pantalla táctil.
- ✓ Permite ingresar a Internet
- ✓ Poseen agenda digital, administración de contactos
- ✓ Permitan leer documentos en distintos formatos, entre ellos los PDFs y archivos de Microsoft Office
- ✓ Debe contar con algún sistema operativo.

Con un teléfono inteligente se puede hacer de todo al mismo tiempo, se puede recibir llamadas, revisar agenda mientras se pueden unos videos en Media Player, o mientras se está sincronizas el dispositivo con otros, y todo esto sin necesidad de interrumpir alguna de las tareas, para no ir tan lejos, es lo mismo que se hace en el computador, se

abren ventanas y todas funcionan al tiempo y no como en un teléfono convencional que si se revisa la agenda se debe dejar de escuchar música para hacerlo.

Tal es el caso de los equipos denominados Smartphone, conocidos también como teléfonos inteligentes, ya que no sólo sirven como dispositivo de comunicación, sino que además son un completo organizador personal.

Para los usuarios que requieren una herramienta portátil que permita realizar diversas tareas similares a las que se puedan llevar a cabo en un computador, además de comunicarse, entonces seguramente el usuario necesitara de un Smartphone.

Una de sus características más destacadas reside en la posibilidad que nos brinda poder instalar programas, mediante los cuales el usuario logra ampliar las capacidades y funcionalidades del equipo, más allá de cómo lo haya entregado el fabricante.

Asimismo, el Smartphone se diferencia del resto de los móviles debido a una serie de características que hacen de él un teléfono inteligente.

Entre las características mencionadas se destacan su excelente acceso y conectividad a Internet, su soporte de clientes de correo electrónico, la eficaz administración de nuestros datos y contactos, entre otras.

Por otra parte, el Smartphone ofrece la posibilidad de lectura de archivos en diversos formatos de acuerdo a las aplicaciones previamente instaladas, incluyendo las más conocidas suites ofimáticas, como es el caso de Microsoft Office.

En cuanto a su diseño, por lo general los Smartphone poseen un tamaño significativamente mayor al de un teléfono móvil convencional, esto se debe a la necesidad de incorporar ciertas características especiales como teclados del tipo Qwerty, pantallas táctiles más grandes de alta definición, entre otras.

CAPÍTULO 4

4 Caso de Estudio

4.1 Caso de estudio del proyecto

El proyecto se centra en el estudio de datos perteneciente a una base de datos de la empresa Bata S.A de la tienda afiliar Timberland, empresa dedicada en la venta de ropa y calzado en Chile y en otros países. Cabe mencionar que las base de dato trabajada en esta tesis, vienen de un proceso ETL previo, por lo cual se centra en el estudio de análisis de los datos resultante. El motor de base datos ocupado es un SQL Server 2008 R2 y para la generación de los cubos se utilizó Analisys Services.

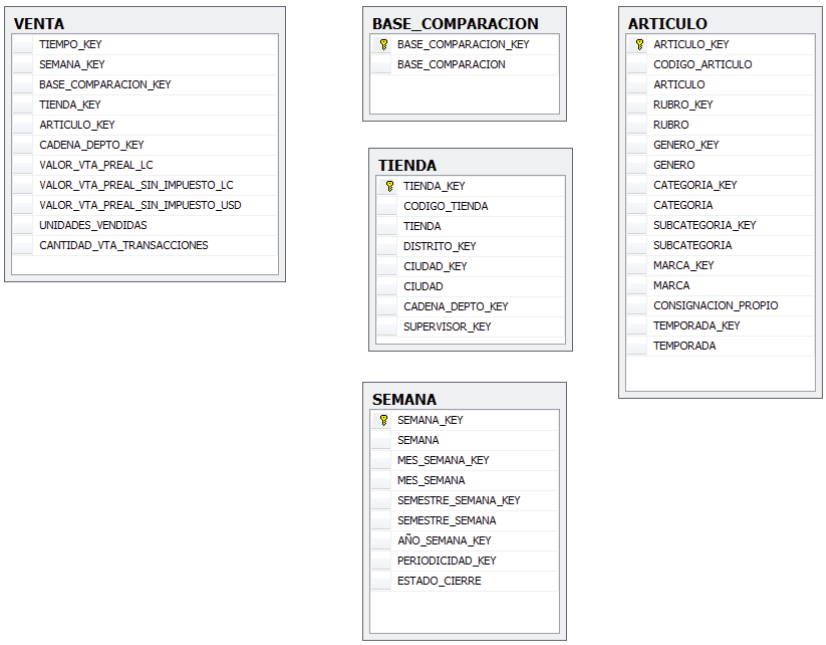


Figura 18 - Base de datos Bata S.A. Fuente: Elaboración propia.

4.2 Cubo análisis OLAP

Un cubo es una unidad de consulta multidimensional, el proceso de construcción consta de 3 pasos

Paso 1: Se debe ubicar la tabla que relaciona las demás, como en la Figura 15 la tabla Ventas es la que relaciona Artículo, Tienda, Semana, Base comparación.

Paso 2: Se debe eliminar las relaciones entre las tablas, y dejar sólo las llaves de las claves primarias ver Figura 15.

Paso 3: Con la herramienta Analisys Service en la opción Cubos al trabajar con tablas existentes se debe escoger la tabla de hechos en este caso se utilizó Ventas y el resto de las tablas se conoce como dimensiones, al generar el cubo esta automáticamente relacionara las tablas existentes.

Con esto se puede procesar el cubo creado y realizar análisis de los datos obtenidos.

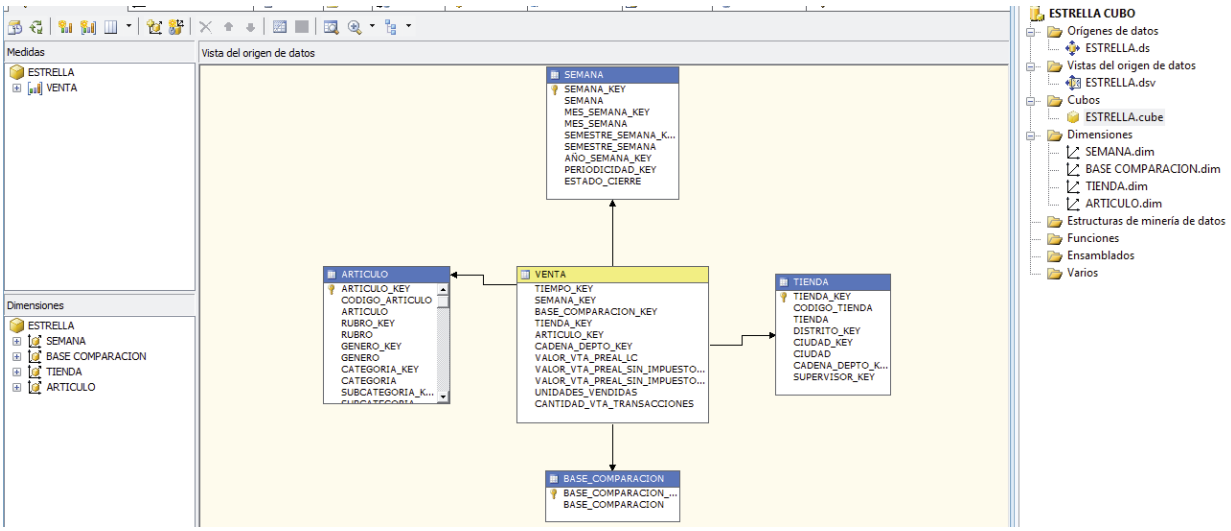


Figura 19 - Ejemplo pasó 3. Fuente: Elaboración propia.

4.3 Tabla de hechos

Una tabla de hechos (o tabla Fact) es una tabla central de un esquema multidimensional ya sea estrella o copo de nieve y que contiene los valores de las medidas de negocio. Cada medida es el resultado de cada intersección de las dimensiones que la definen, estas dimensiones son reflejos de sus correspondientes tablas de dimensiones que rodean la tabla de hechos y que se relacionan con ella.

4.3.1 **Medidas tabla de hechos**

Las medidas comúnmente utilizadas en una tabla de hecho son los aditivos, es decir aquellas medidas que pueden ser sumadas como la cantidad de un producto vendido son medidas numéricas que pueden calculares con la suma de varias cantidades de tablas o también el cálculo total de cantidad de producto por el precio, etc. En consecuencia, los hechos a almacenar en una tabla de hechos representan casi siempre a valores numéricos.

4.3.2 **Cardinalidad tabla de hechos**

Las tabla de hechos pueden contener un gran número de filas, a veces cientos de millones de registros cuando contienen uno o más años de historia de una organización, esta cardinalidad estará acotada superiormente por una cardinalidad de las tablas dimensionales. Por ejemplo si se tiene una tabla de hechos TH con tres dimensiones D1, D2, D3, el número máximo de elementos que tendrá la tabla de hechos TH será:

Card(TH) = Card(D1) x Card(D2) x Card(D3)
Donde Card(x) es la cardinalidad de la table x

Tabla 5 - Cardinalidad de una tabla x.

Estas cardinalidades no son fijas, debido si una de las dimensiones se refiere a los clientes de la empresa, por ejemplo, cada vez que se dé un alta un nuevo cliente estará aumentando la cardinalidad de la tabla de hechos. Unas de las dimensiones suele ser el tiempo, este puede medirse de distintas formas (Año, Semestre, Mes, Semana, Días, Horas), pero lo cierto es que transcurre continuamente, y para que el sistema funcione se deben añadir registros periódicamente a la tabla de esta dimensión (tabla de tiempos) y esto también produce un aumento de la cardinalidad de la tabla de hechos, esta es la principal causa de que las tablas de hechos lleguen a tener una cantidad de registros del orden de millones de elementos.

4.3.3 **Granularidad**

Una característica importante que definen una tabla de hechos es el nivel de granularidad de los datos que en ella se almacenan, entendiéndose por granularidad el nivel de detalle de dichos datos, es decir, la granularidad de la tabla de hechos representa el nivel más atómico por el cual se definen los datos, por ejemplo no es lo mismo contar el tiempo por horas (grano fino) que por semanas (grano grueso).

Cuando la granularidad es mayor, es frecuente que se desee disponer subtotales parciales, es decir, si tenemos una tabla de hechos con las ventas por días, podría interesar disponer de totales semanales y/o mensuales, estos datos se pueden calcular haciendo sumas parciales, además es frecuente añadir a la tabla de hechos registros donde se almacenan dichos cálculos para no tener que repetirlos cada vez que se requieran y mejorar el rendimiento de la aplicación.

4.3.4 **Agregación**

Es un proceso de cálculo por el cual se resumen datos de los registros de detalle. Esta operación consiste normalmente en el cálculo de totales dando lugar a medidas de grano grueso. Cuando se resumen los datos, el detalle ya no está directamente disponible para el analista, ya que este se elimina de la tabla de hechos.

Esta operación se realiza típicamente con los datos más antiguos de la empresa con la finalidad de seguir disponiendo de dicha información (aunque sea resumida) para poder eliminar registros obsoletos de la tabla de hechos para liberar espacio.

4.4 Arquitectura

En la figura siguiente se muestra la arquitectura implementada para este proyecto. Como se comentó anteriormente se diseñó un proyecto móvil capaz de comunicarse con un Servlet creado en java y este a su vez se comunica con un cubo creado en un motor de base de datos SQL SERVER 2008.

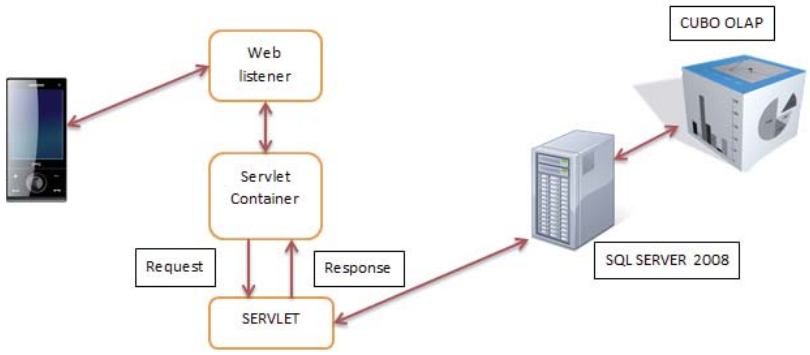


Figura 20 - Arquitectura proyecto móvil. Fuente: Elaboración propia.

La arquitectura desarrollada se comunica a través de un Servlet, es te envía una petición al motor de base de datos, el cual ejecuta una Query hacia el cubo creado, la repuesta del cubo consultado devuelve como resultado, datos que son procesado por el Servlet, creando un archivo XML. Este archivo, el dispositivo móvil lo descarga y lo interpreta, graficando los datos que serán mostrados al usuario final.

CAPÍTULO 5

5 Discusión de Resultados

5.1 Análisis de datos

En este capítulo se centra en mostrar los resultados obtenidos tanto en la creación y el estudio de análisis de datos (inteligencia de negocio), un Servlet (transfiere datos) y un dispositivo móvil que permita representar los datos. Para la primera se trabajó con dos modelos un esquema estrella y el otro copo de nieve obteniendo información del proceso que el cubo genera en ambas.

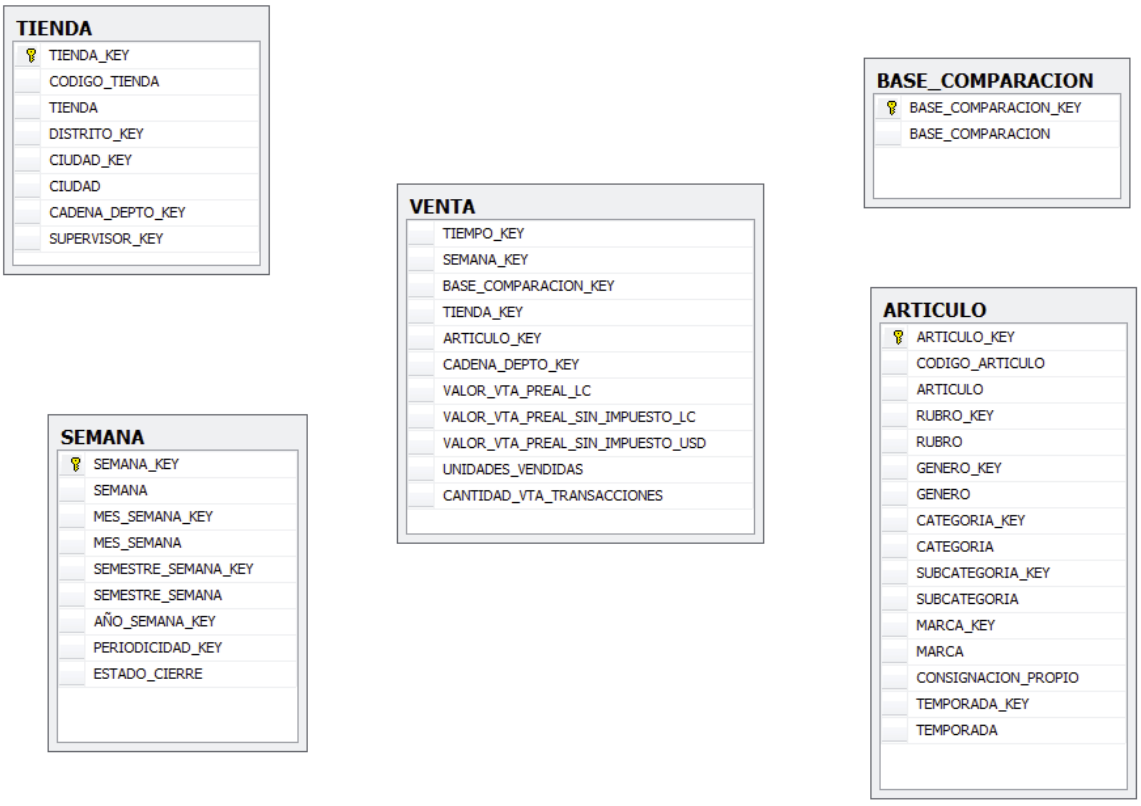


Figura 21 - Modelo de tabla Estrella. Fuente: Elaboración propia (Generado en SQL Server 2008).

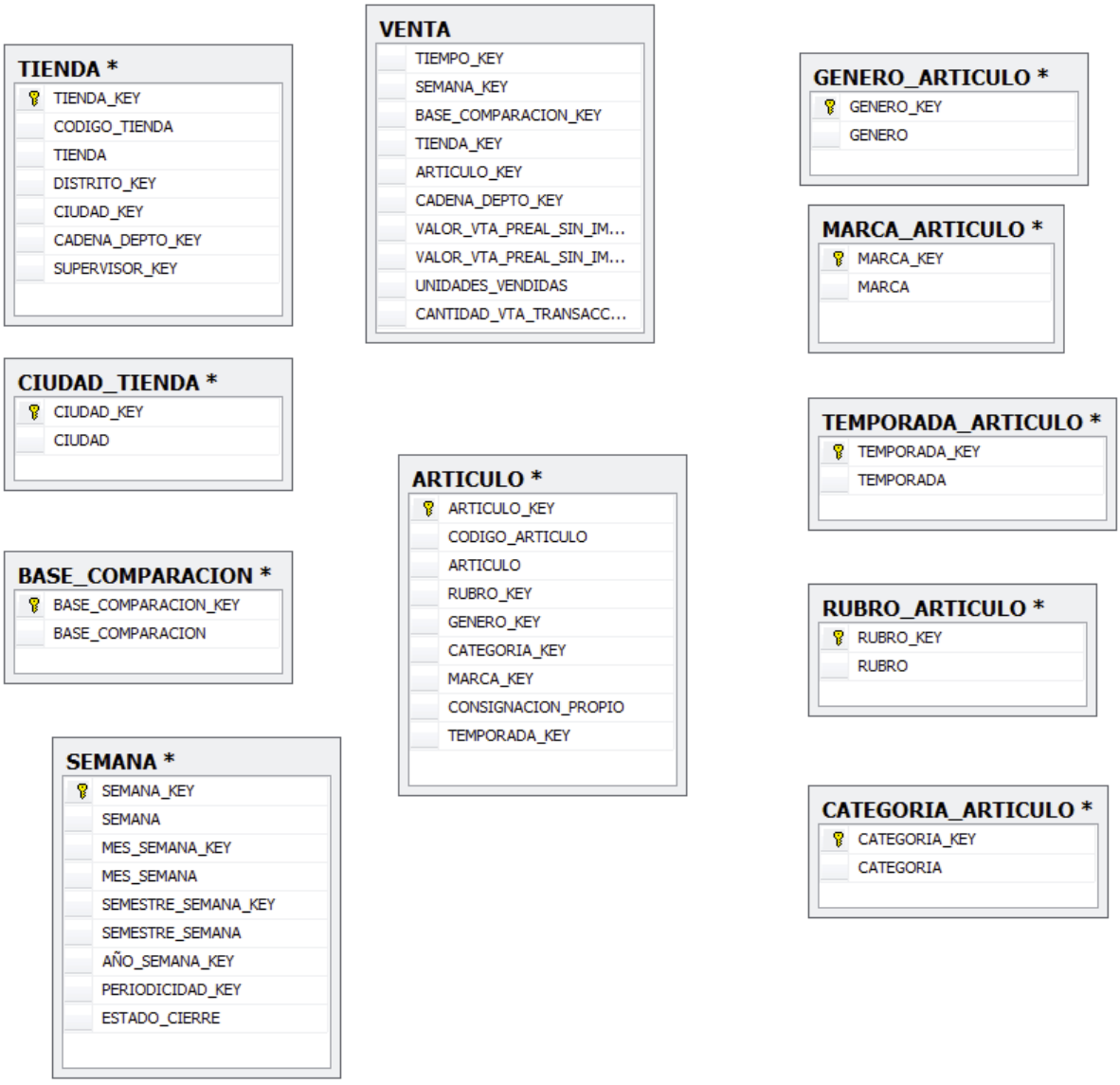


Figura 22 - Modelo de tabla Copo de nieve. Fuente: Elaboración propia (Generado en SQL Server 2008).

Recordemos que los cubos permiten analizar datos críticos desde cualquier ángulo, en cualquier combinación. Gracias a esta técnica es posible realizar análisis de criterio del usuario, desde lo más sencillo a los más complejos.

Para ello se procesó el cubo con la información de las tabla existentes dependiendo del modelo de lo datos (estrella o copo de nieve) cada proceso toma un tiempo diferente de carga, tiempo que es desigual debido a la desagregación de tablas para el caso de copo de nieve, aunque cabe destacar la diferencia entre cargas es de 3 segundos de diferencia teniendo en cuenta que son más de tres mil registros.

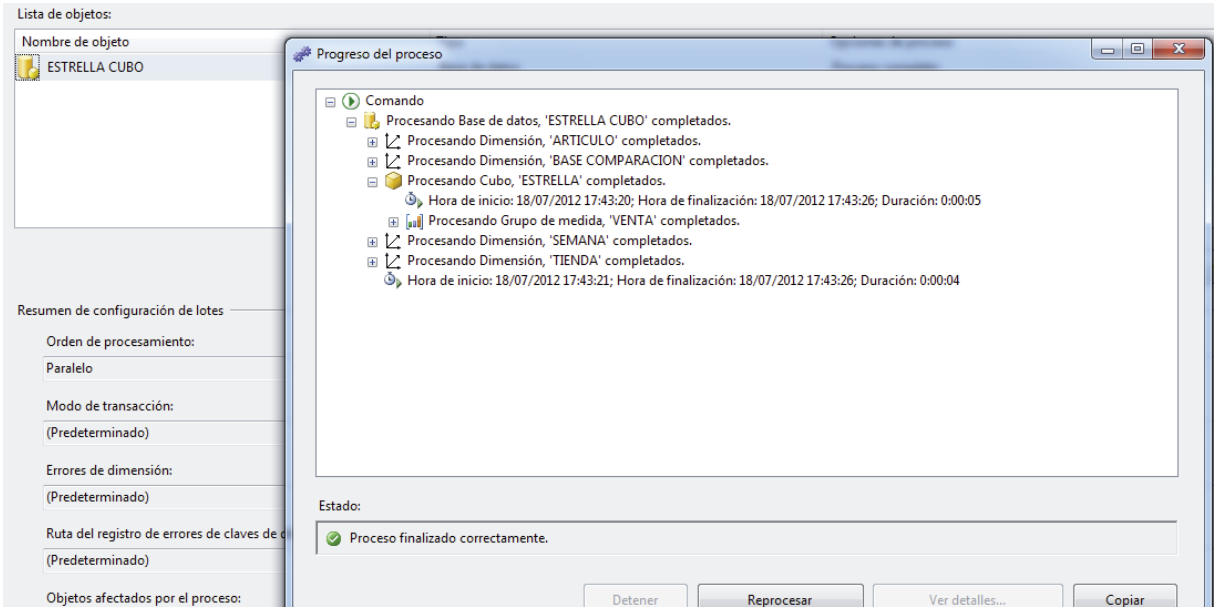


Figura 23 - Proceso modelo Estrella. Fuente: Elaboración propia (Generado en SQL Server 2008).

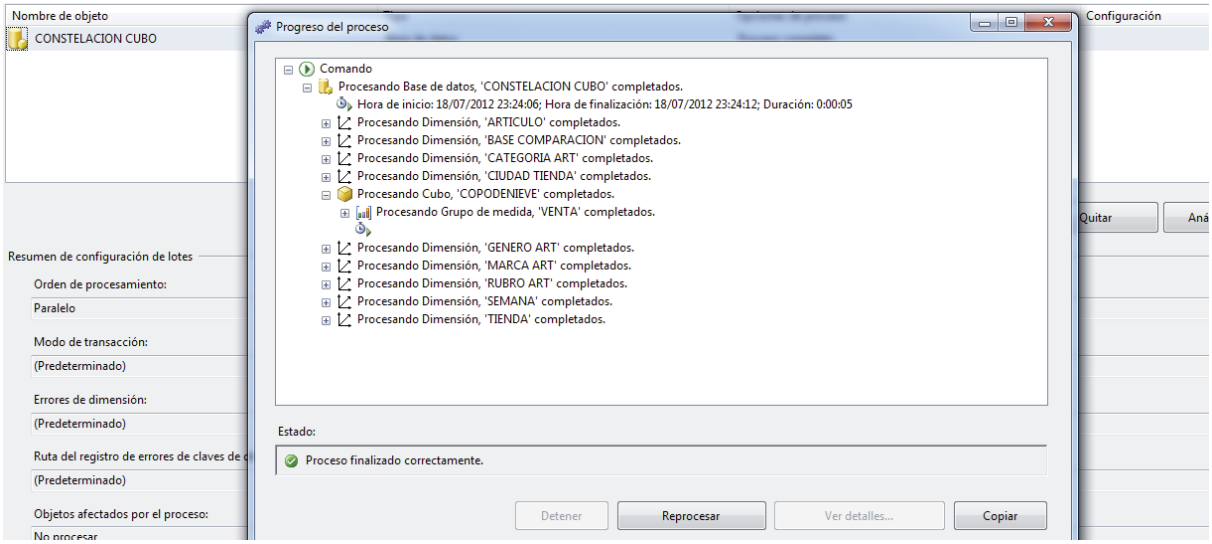


Figura 24 - Proceso modelo Copo de nieve. Fuente: Elaboración propia (Generado en SQL Server 2008).

En las imágenes 23 y 24 se muestra el proceso de las distintas dimensiones existentes para cada cubo, y el mensaje de confirmación exitosa.

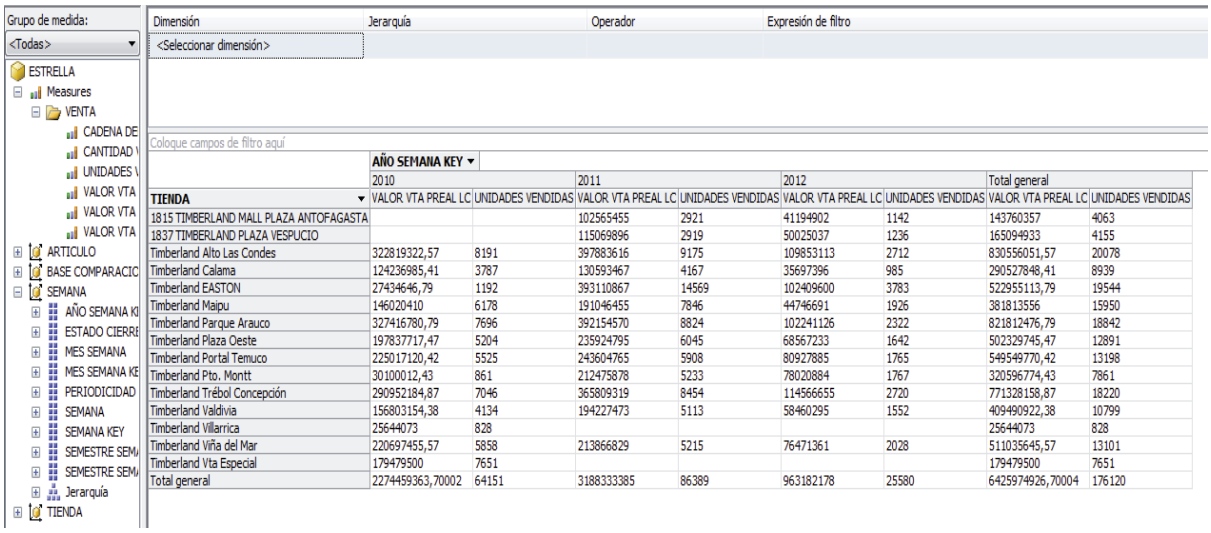


Figura 25 - Consulta generada a través de Analysis Service modelo Estrella. Fuente: Elaboración propia.

Al generar los procesos de dichos cubos se podrá realizar diferentes tipos de consulta correspondiente, estas consultas se pueden desarrollar a modo de drag and drop (arrastrar y soltar) en la misma aplicación Analisys Service o también a través de Excel, este ultimo de pude conectar a los servicios de red de SQL SERVER.

En la figura 26 se muestra una consulta generada por la herramienta Analysis Service donde se consulta el valor de la venta real y las unidades vendidas de todas las tiendas Timberland durante los años 2010, 2011, 2012.

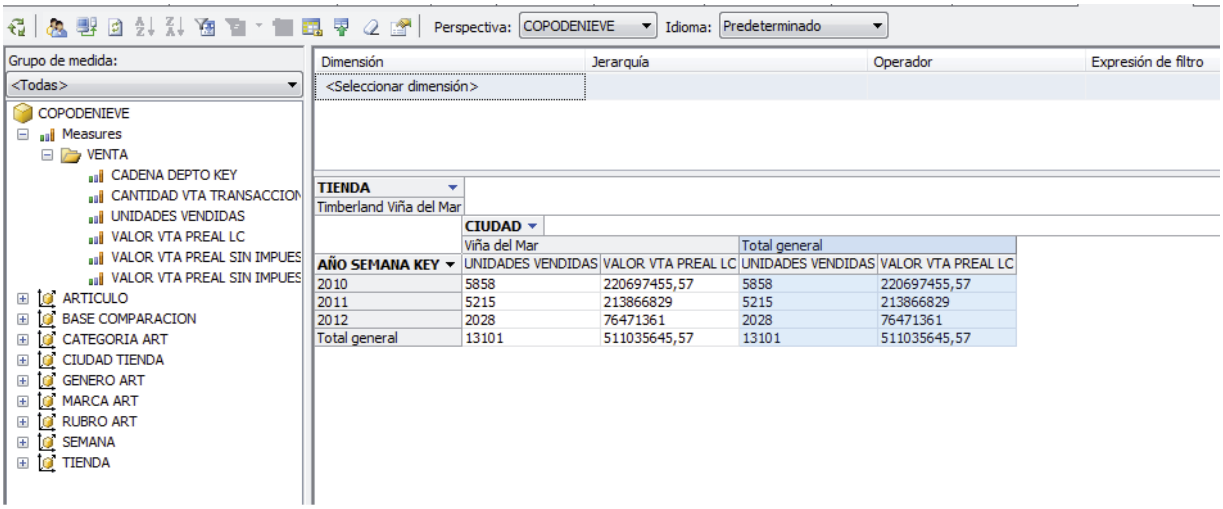


Figura 26 - Consulta generada a través de Analysis Service modelo Copo de nieve. Fuente: Elaboración propia.

En la figura 27 se muestra la consulta de unidades vendidas y el valor de las ventas reales de las distintas ciudades pertenecientes a la tienda Timberland.

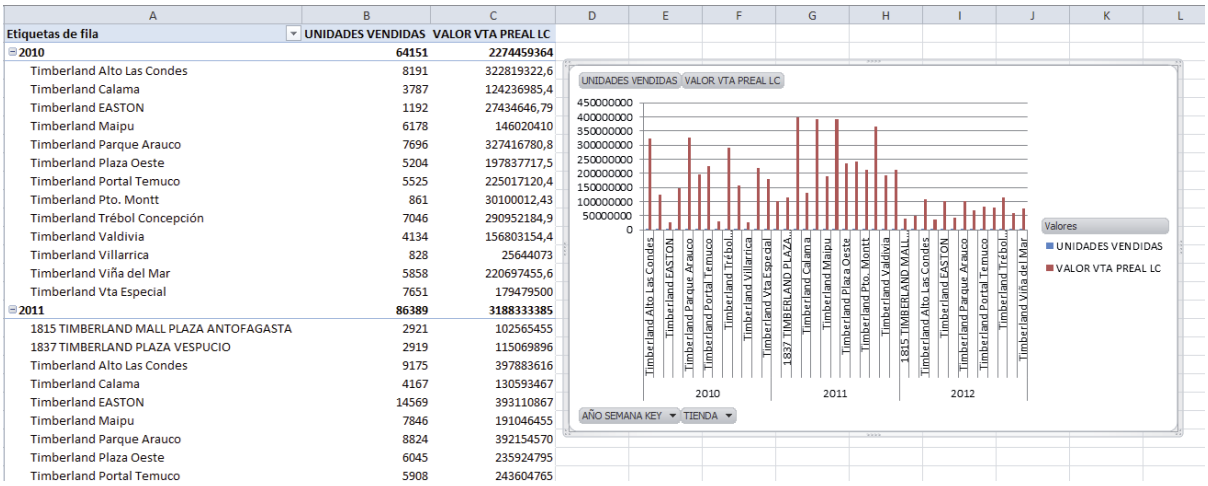


Figura 27 - Consulta generada a través de Excel modelo Estrella. Fuente: Elaboración propia.

En la figura 28 se muestra el mismo ejemplo que la figura 27, pero el resultado del cubo es generado a través de la aplicación Excel el cual se puede trabajar directamente con gráficos haciendo más amigable para el usuario y además haciendo más fácil la comprensión de los resultados y la toma de decisiones.

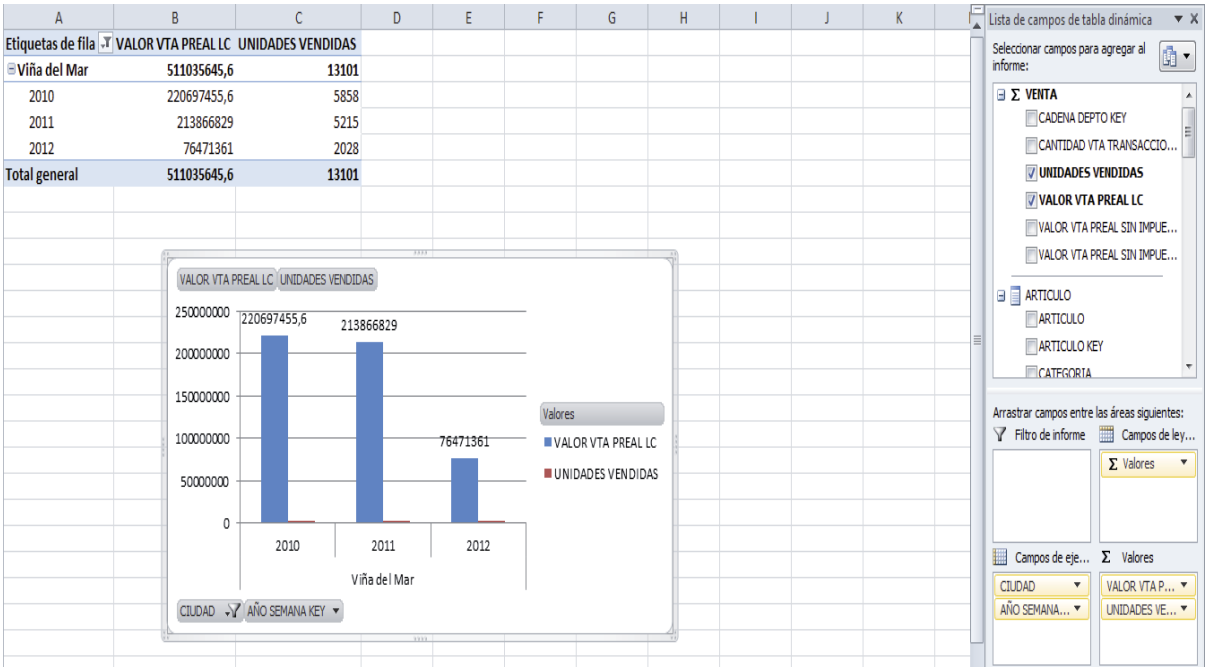


Figura 28 - Consulta generada a través de Excel modelo Copo de nieve. Fuente: Elaboración propia.

La figura 28 es un ejemplo visual generado en Excel y es la misma consulta realizada en la figura 26 a través de Analysis Service.

Para la generación del Servlet se trabajó con la librería OLAP4J que permite realizar las consultas directamente al cubo a través de XMLA que es un estándar de análisis de datos, tales como OLAP y minería de datos.

XMLA se basa en estándares de la industria tales como XML, SOAP, HTTP. Este además consta con 2 métodos SOAP que son execute y discover.

- Execute

  Ejecutar método tiene dos parámetros:

  Comando - el comando a ejecutar. Puede ser MDX, DMX o SQL.

  Propiedades: XML lista de propiedades de comando como el tiempo de espera, el nombre del catálogo, etc

  El resultado de la orden de ejecución podría ser conjunto de datos multidimensionales o conjunto de filas tabular.

- Discover

  Método Discover fue diseñado para modelar todos los métodos de detección de posibles OLEDB incluyendo varios conjuntos de filas de esquema, propiedades, palabras clave, etc Discover método permite a los usuarios especificar lo que debe ser descubierto y las posibles restricciones o propiedades. El resultado del método Discover es un conjunto de filas.

```
// We must use the XMLA driver.
Class.forName("org.olap4j.driver.xmla.XmlaOlap4jDriver");
// This code is for Java 5. With Java 6, you can directly
// unwrap the underlying connection with the .unwrap() call.
OlapConnection connection =
(OlapConnection) DriverManager.getConnection(

// This is the SQL Server service end point.
"jdbc:xmla:Server=http://example.com/olap/msmdpump.dll"

// Tells the XMLA driver to use a SOAP request cache layer.
// We will use an in-memory static cache.
+ ";Cache=org.olap4j.driver.xmla.cache.XmlaOlap4jNamedMemoryCache"

// Sets the cache name to use. This allows cross-connection
// cache sharing. Don't give the driver a cache name and it
// disables sharing.
+ ";Cache.Name=MyNiftyConnection"
```

```
// Some cache performance tweaks.
// Look at the javadoc for details.
+ ";Cache.Mode=LFU;Cache.Timeout=600;Cache.Size=100",

// XMLA is over HTTP, so BASIC authentication is used.
"username",
"password" );
// We can execute a query. MDX of course.
CellSet set = connection.createStatement().executeOlapQuery(
"SELECT {} ON COLUMNS FROM CUBE");
```

Tabla 6 - Ejemplo de conexión con OLAP4J.

Además se debe crear el servicio IIS (internet information service) que es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows, este proporcionara las herramientas necesarias para administrar un servidor seguro.

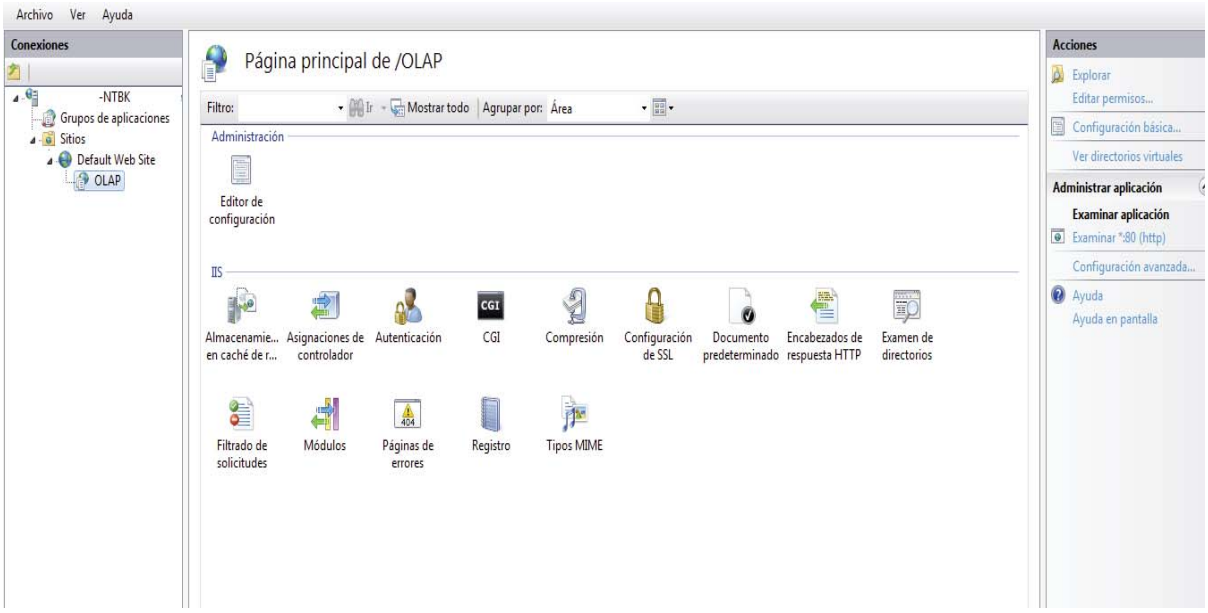


Figura 29 - Configuración IIS para Analysis service. Fuente: Elaboración propia (Generado en SQL Server 2008).

Con los pasos anteriormente descrito se podrá programar en Cocoa el cual se conectara al Servlet permitiendo enviar y recibir peticiones de consulta, generando gráficos en los dispositivos móviles para las tomas de decisiones.

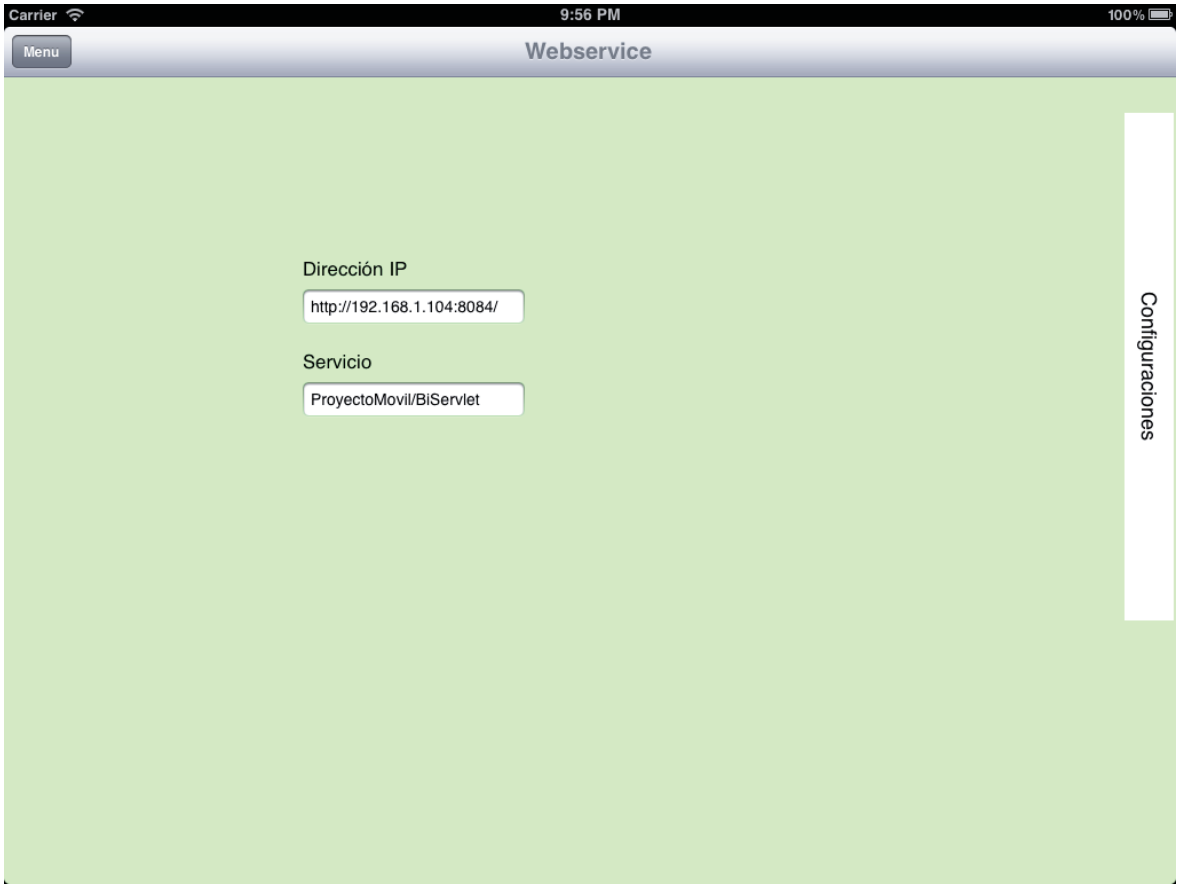


Figura 30 – Pantalla de inicio de la aplicación en IPad. Fuente: Elaboración propia.

En la figura 30 se muestra la pantalla de inicio, en él se debe ingresar la dirección ip de la máquina servidor BI e ingresar el nombre del servicio, con estos datos se accede al sistema.

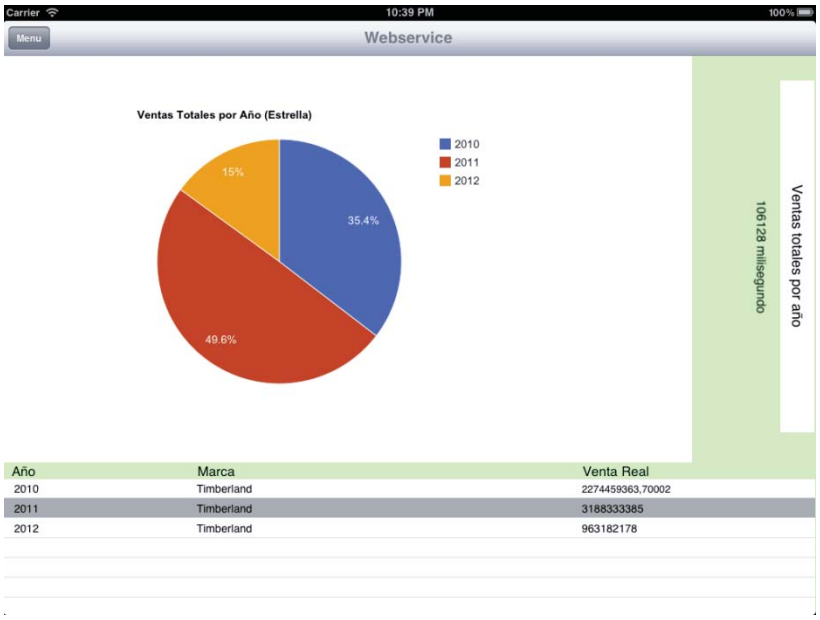


Figura 31 – Ventas totales por año (Estrella). Fuente: Elaboración propia.

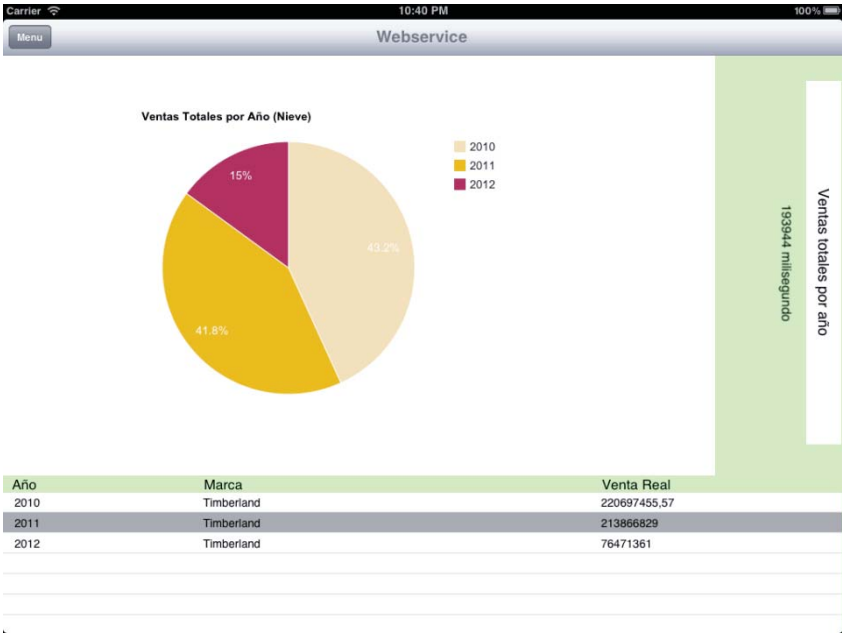


Figura 32 – Ventas totales por año (Nieve). Fuente: Elaboración propia.

En las figuras de tortas 31 y 32 se muestran las ventas totales realizadas dentro de los años 2010 hasta el 2012, para la primera se realizó conectándose al cubo estrella mientras la segundo al cubo copo de nieve.

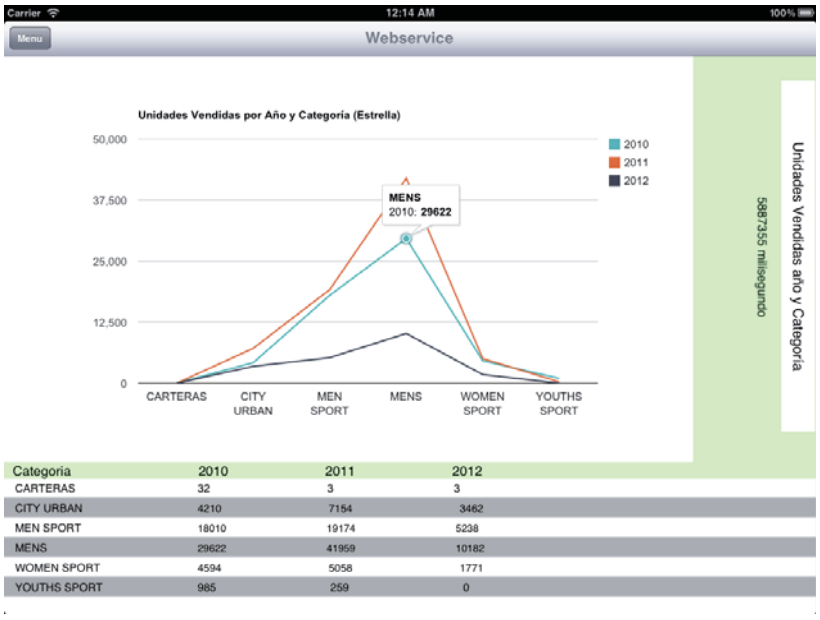
Año/semestre	Ciudad	Ventas
2010/Primer Semestre	Calama	49911147
2010/Primer Semestre	Concepción	125058999
2010/Primer Semestre	Stgo - A. Las Condes	131817199
2010/Primer Semestre	Stgo - Maipu	28471474
2010/Primer Semestre	Stgo - Pque. Arauco	129481025
2010/Primer Semestre	Stgo - Pza. Oeste	78447701
2010/Primer Semestre	Temuco	98693905

Figura 33 – Unidades ventas por semestre (Estrella). Fuente: Elaboración propia.

Año/semestre	Ciudad	Ventas
2010/Primer Semestre	Calama	49911147
2010/Primer Semestre	Concepción	125058999
2010/Primer Semestre	Stgo - A. Las Condes	131817199
2010/Primer Semestre	Stgo - Maipu	28471474
2010/Primer Semestre	Stgo - Pque. Arauco	129481025
2010/Primer Semestre	Stgo - Pza. Oeste	78447701
2010/Primer Semestre	Temuco	98693905

Figura 34 – Unidades ventas por semestre (Nieve). Fuente: Elaboración propia.

En las figuras de barra 33 y 34 se muestran las unidades vendidas por semestre en cada ciudad desde el año 2010 hasta el año 2012, para la primera se realizo la conexión al cubo estrella y para la segunda al cubo copo de nieve.



Categoria	2010	2011	2012
CARTERAS	32	3	3
CITY URBAN	4210	7154	3462
MEN SPORT	18010	19174	5238
MENS	29622	41959	10182
WOMEN SPORT	4594	5058	1771
YOUTHS SPORT	985	259	0

Figura 35 – Unidades vendidas por año y categoría (Estrella). Fuente: Elaboración propia.

Categoria	2010	2011	2012
CARTERAS	32	3	3
CITY URBAN	4210	7154	3462
MEN SPORT	18010	19174	5238
MENS	29622	41959	10182
WOMEN SPORT	4594	5058	1771
YOUTHS SPORT	985	259	0

Figura 36 – Unidades vendidas por año y categoría (Nieve). Fuente: Elaboración propia.

Para las figuras de líneas 35 y 36 se muestra las unidades vendidas por año y categorías realizadas durante el año 2010 hasta el año 2012, la primera se realizó conectándose al cubo estrella, mientras la segunda se consultando al cubo copo de nieves.

CAPÍTULO 6

Conclusiones y Trabajo Futuro

En la presente tesis de grado se desarrolló el estado del arte, demostrando la utilidad y el enfoque que las empresas están dando en la utilización de software móvil para la toma de decisiones.

Las empresas que actualmente manejan un volumen de datos mayor, optan por implementar el uso de la Inteligencia de Negocios manejando convenientemente la transformación de datos en información. Para ello se analizó y explicó sobre la utilización de cubos los cuales permiten buscar datos con rapidez y tiempo de respuesta uniforme, independientemente de la cantidad de datos en el cubo o la complejidad del procedimiento de búsqueda.

Se explicó el uso de esquemas estrellas la cual se compone de una tabla de hechos y una tabla adicional por cada dimensión, un esquema copo de nieve donde se refleja la organización jerárquica de las dimensiones y el esquema constelación donde los dos esquemas anteriores pueden generalizarse con la inclusión de distintas tablas de hechos que comparten todas o algunas de las dimensiones.

Gracias a la utilización de Analysis services (SSAS) en este proyecto se pudo diseñar, crear y administrar estructuras multidimensionales utilizando como objeto de estudio la base de datos Bata S.A. lo cual se visualizó mediante la herramienta gráficos que permiten entender, analizar y tomar una decisión antes los resultados obtenidos.

Mediante los Servlets desarrollados en java (J2EE) se puede enlazar y comunicar información entre el cubo y el dispositivo móvil siendo el objetivo del proyecto, permitiendo a los hombres y mujeres del negocio empresarial tomar decisiones desde cualquier parte.

Durante el desarrollo de esta tesis y en conjunto con la herramienta SQL Server, se modeló 2 casos de pruebas en el primero se utilizó un modelo de cubo estrella compuesto por una tabla de echo y una tabla adicional por cada dimensión y en el segundo un modelo de copo de nieve donde se refleja un orden jerárquico de las dimensiones, el cual a través de tiempos de respuesta, los cubo fueron medidos para demostrar la utilización de estos en el proyecto final.

Se construye una aplicación java que permite la conexión a los distintos cubos, dicha aplicación además permitió interpretar los resultados enviándolos al aplicativo móvil para ser graficado.

Los resultados obtenidos permitieron dar a conocer que el modelo estrella, responde con una tasa menor de tiempo a pequeños y grandes volúmenes de datos eso si cabe destacar que las diferencias en respuesta solo fueron en milisegundos.

Además de la utilización de SQL Server (SSAS) y la creación de un software de comunicación, se desarrolló una aplicación para iPhone y/o iPad mediante la herramienta Xcode, la cual permitió leer los archivos creados en XML (Java) y plasmarlo mediante gráficos para su interpretación y tomas de decisiones.

Por otra parte el desarrollo de aplicaciones para iPhone e iPad va creciendo debido a que las empresas están dando otro enfoque a las tecnologías Smartphone o Tablet, sacando el rendimiento real a los equipos.

Con el estudio realizado se da a conocer como una buena propuesta el uso del cubo estrella debido al tiempo obtenido en respuesta; como fue dicho anteriormente la diferencia fue de milisegundos. Uno de los problemas grandes en esta tesis fue la comunicación entre las librerías OLAP de java y los cubos OLAP debido a la poca información disponible encontrada en libro y en la web. Gracias al estudio y la investigación se pudo establecer la comunicación permitiendo obtener respuestas desde la base de datos.

Por otro lado, los resultados obtenidos en java permitieron interactuar con la aplicación móvil desarrollada obteniendo buenos resultados. En consecuencia, es posible aseverar que se cumplieron todos los objetivos propuestos.

Para un trabajo futuro se espera que se trabaje en la implantación de KPI (Indicadores Claves de Desempeño), trabajar a través de webservices que permitan conectarse con distintos equipos móviles (no sólo iPhone y/o iPad), pudiendo este proyecto ser aplicado en otras empresas.

Finalmente, podemos establecer que los Smartphone no sólo pueden ser usados como celulares para hablar, ya que cuentan con una infinidad de utilidades y se han desarrollado diversas aplicaciones, para obtener el máximo de utilidad de estos, pudiendo dejar casi de lado a los computadores. Por lo mismo, es importante el desarrollo de aplicaciones de utilidad, más allá de las aplicaciones de juegos, para ver el estado del tiempo, acceso a Internet, si no que se puede contar ya sea en los Smartphone y en los Tablet o iPad de una aplicación que permita estar conectado con la información de la empresa relacionado con las ventas del negocio, reportes, ver tendencias del negocio lo que sin duda es relevante para los usuarios que no se encuentran siempre en la oficina y necesitan acceder a información para la toma de decisiones, como por ejemplo, si es el momento de lanzar una promoción si las ventas están bajas o adelantar la salida del producto, o simplemente consultar como están las ventas o como se han comportado las ventas en comparación con el periodo anterior. De esta forma el celular se convierte en una herramienta de trabajo conectado a la información de la compañía que hoy en día resulta de vital importancia.

Referencias

[1]Documento en línea IOS Developer Library, 14 Nov 2011, http://developer.apple.com/library/ios/navigation/

[2] Rodrigo Frez, curso de Inteligencia de Negocios, programa de Postgrado en Gestión TI, Universidad de Chile 2009.

[3] José Luis Martí Lara, curso de Inteligencia de Negocios, programa de Magister en Ingeniería Informática, Pontificia Universidad Católica 2010.

[4] Learning Cocoa with Objective-C autor James Duncan Davidson Copyright © 2002, 2001 O'Reilly & Associates, Inc. All rights reserved.

[5] Core Servlets and JavaServer Page Marty Hall – Larry Brown Java 2 Platform, Eterprise Edition Series.

[6] Kimball, R.:"TheDatawarehouseToolkit". John Wiley& Son, Inc., 1996.

[7] Kortnik, M. Moody, D.: "FromEntitiestoStars, Snowflakes, Clusters, Constellations and Galaxies: A Methodologyfor Data WarehouseDesign". ER'99 Industrial Track, France, 1999.

[8] Marotta, A. Peralta, V.: "Diseño de Data Warehouses: Un Enfoque Basado en Transformación de Esquemas". Info-UY'99,Uruguay,1999.

[9] Marotta, A.: "Data WarehouseDesign and MaintenancethroughSchemaTransformations". Master Thesis. Advisor: Raúl Ruggia. Pedeciba, Universidad de la República, Uruguay, 2000.