UNIVERSIDAD CATÓLICA DEL MAULE
Facultad de Ciencias de la Ingeniería
Escuela de Ingeniería Civil Informática

Profesor Guía
Dra. Angélica Urrutia S.

"IMPLEMENTACIÓN DE UN ALMACÉN DE DATOS PARA UNA BASE DE DATOS DB2 USANDO INSTRUCCIONES SQL: UNA SOLUCIÓN ROLAP"

Jessica Figueroa González

Tesis para optar al Título Profesional de

Ingeniero Civil Informático

Talca, Abril 2007

UNIVERSIDAD CATÓLICA DEL MAULE
FACULTAD DE CIENCIAS DE LA INGENIERÍA
ESCUELA INGENIERÍA CIVIL INFORMÁTICA

TESIS PARA OPTAR AL

GRADO DE
LICENCIADO EN CIENCIAS DE LA INGENIERÍA
TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

"IMPLEMENTACIÓN DE UN ALMACÉN DE DATOS PARA UNA BASE DE DATOS DB2 USANDO INSTRUCCIONES SQL: UNA SOLUCIÓN ROLAP"

JESSICA FIGUEROA GONZÁLEZ

COMISIÓN EXAMINADORA	**FIRMA**
PROFESOR GUÍA DRA. ANGÉLICA URRUTIA SEPÚLVEDA	_____
PROFESOR COMISIÓN YESSICA GÓMEZ GUTIÉRREZ	_____
PROFESOR COMISIÓN RODOLFO VILLARROEL ACEVEDO	_____
NOTA FINAL DE TÍTULO	_____

TALCA, ABRIL DE 2007

“Con todo mi amor a mis padres Luis y Carmen”

AGRADECIMIENTOS

Al finalizar esta etapa miro hacia atrás y me doy cuenta que hubieron muchas personas que de una u otra forma ayudaron a que cumpliera mi objetivo, a todas estas personas quiero agradecer en este momento tan importante….

En primer lugar a Dios por darme las fuerzas que siempre necesité. A mis padres que con gran esfuerzo me dieron la oportunidad de estudiar, hermanos y sobrinos que son el pilar fundamental y lo más importante en mi vida, por sobre todo a mi hermano Oscar por su total apoyo y ayuda.

A Marcelo por su amor, comprensión y apoyo incondicional en estos años, gracias por estar ahí cuando te necesité y por tus palabras de aliento cada vez que creí que no lo podía lograr.

A mi amigo Sebastián que me ayudo siempre, gracias por tu paciencia. A mis amigas Naty, Marce, Ivonne y Sole, como olvidar todas esas largas noches de estudio, los buenos momentos y locuras de todos estos años, las quiero mucho.

A la Dra Angélica Urrutia por guiar este trabajo.

A cada uno de Uds gracias, espero estén conmigo en esta nueva etapa que comienza…..

RESUMEN

Actualmente se puede observar que para las empresas es cada vez más importante poder realizar un análisis oportuno de la información que dará soporte al proceso de toma de decisiones dentro de éstas.

Una de las soluciones más aplicadas en esta área son los Almacenes de Datos, es por esta razón que este trabajo se aboca a construir una solución de este tipo implementando un Data Warehouse utilizando el administrador de base de datos relacional DB2, ocupando sólo instrucciones SQL que este administrador proporciona y que simulan cubos multidimensionales indispensables para el proceso de análisis de información sin tener que recurrir a herramientas o nuevas tecnologías que generen estos cubos.

El almacén de datos debe ser capaz de responder a ciertos indicadores que son relevantes para el proceso de toma de decisiones y de acreditación universitaria de una institución del país, estos indicadores se determinaron durante un trabajo realizado en conjunto con académicos y estudiantes de la institución.

A través del desarrollo de este trabajo además se podrán conocer cuales son las potencialidades de DB2 como administrador de base de datos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

“Introducción”

1. INTRODUCCIÓN

Para las organizaciones la información es poder, y que ésta proporcione la utilidad necesaria dependerá de donde se almacena toda la información que necesita, como está organizada y de que forma puede hacerla disponible a las distintas personas, empresas, etc.

La importancia de una buena información puede ser vista como la diferencia en valor entre una decisión correcta y una decisión equivocada, en donde la decisión está basada en esa información. Mientras más grande sea esa diferencia entre decisión correcta y errónea, mayor será la importancia de contar con una buena información [BIT2005].

Para tener completamente automatizada a la empresa es necesaria una gran infraestructura en tecnología que soporta sistemas de información. Este crecimiento tecnológico tiene distintos orígenes, que van desde la implementación, crecimiento, ampliación, integración, etc. Las condiciones actuales de los mercados han provocado la necesidad de tecnología cada vez más avanzada para responder a las peticiones muy particulares de información.

En estos momentos la información fluye en todos los niveles de la organización con diferentes fines (comunicación, control, administración, evaluación, etc.) independientemente de los puestos. Las empresas están entendiendo que los niveles directivos tienen una gran responsabilidad al tomar decisiones, pues el impacto que generan recae sobre toda la organización, pero también existen más personas que toman decisiones y, a pesar de que éstas no tienen un impacto global, deben ser también correctas y oportunas, pues ciertos grupos dependen de las mismas.

Directores, gerentes, supervisores, jefes, todos aquellos que toman decisiones deben tener suficiente información para apoyarse en su trabajo cotidiano, el lugar que ocupen en la pirámide organizacional se vuelve secundario cuando el enfoque es hacia el manejo de procesos y todos los puestos tienen cierta relación y dependencia entre sí.

De modo general en una pirámide organizacional, los requerimientos informativos se dividen en 3 partes [CAS2001]:

• Información Estratégica
• Información Táctica
• Información Técnico Operacional.

Para tomar decisiones en cualquiera de los niveles de la pirámide es necesario hacer gestión sobre los datos de la organización, que por lo general, los sistemas operacionales, la presentan en bruto, además esta gestión debe ser acorde a las necesidades de la empresa y dependiendo de lo que se está dispuesto a invertir para realizarla.

Hay distintas empresas que ofrecen una variada gama de productos para dar soluciones de negocio y poder obtener así información que dé soporte a la toma de decisiones, entre estos productos se encuentran los Administradores de Bases de Datos. Hoy en día existe una fuerte disputa en este mercado entre Oracle e IBM en donde cada uno de estos pretende demostrar su liderazgo en este tipo de soluciones. Para el desarrollo de la presente tesis se optó por trabajar con el administrador de base de datos DB2 de IBM.

Es por esto que el principal objetivo será diseñar e implementar un almacén de datos como una solución ROLAP usando el modelo relacional e instrucciones SQL, que permita transformar una base de datos operacional a una orientada al proceso de análisis de datos, la que podrá ser vista como un Data Warehouse y apoyar el proceso de toma de decisiones dentro del marco en que está inmersa esta tesis. Además de presentar un material que muestre con un cierto nivel de profundidad las potencialidades de DB2 que es el motor de base de datos escogido.

Por lo tanto, este trabajo se limita a desarrollar un almacén de datos que facilite el proceso de toma de decisión, utilizando como datos fuente parte de la base de datos de la escuela de ingeniería de una Universidad de nuestro país, y que responde a los requerimientos de información solicitados por dicha área.

1.1. OBJETIVOS

- **Objetivo General**

Diseñar e implementar un almacén de datos como una solución ROLAP usando el modelo relacional e instrucciones SQL de la Base de Datos DB2 de IBM.

- **Objetivos Específicos**

1. Conocer las potencialidades del Administrador de Base de Datos DB2 para soluciones de gestión de datos usando instrucciones SQL para ROLAP.
2. Diseñar una solución de gestión de datos, utilizando una plataforma de base de datos ROLAP en DB2 de IBM.
3. Implementar dado una base de datos fuente las instrucciones SQL para ROLAP al caso en cuestión.

1.2. ORGANIZACIÓN DEL DOCUMENTO

El documento está organizado como sigue:

En el segundo capítulo se presenta el marco teórico que servirá de base para el desarrollo de la tesis, específicamente cuenta con una descripción del Administrador de Base de Datos que se ocupó, un breve estudio del concepto de Almacén de Datos y finalmente una descripción de los Sistemas OLAP.

En el tercer capítulo se presenta un completo análisis de los operadores CUBE y ROLLUP que serán parte fundamental de la implementación del caso en estudio.

En el cuarto capítulo se expone la metodología ocupada para el diseño e implementación del Almacén de Datos para el caso en estudio, se define el problema a solucionar, se definen los requisitos del usuario, se realiza el análisis del modelo de datos fuente, para posteriormente llevar a cabo la implementación del Data Warehouse. Finalmente, en el quinto capítulo se dan a conocer las conclusiones del presente proyecto de tesis.

Capítulo 2

“Marco Teórico”

2. MARCO TEÓRICO.

2.1. DB2 UNIVERSAL DATABASE (DB2 UDB).

En los últimos tiempos se ha escuchado hablar sobre una nueva base de datos llamada "DB2 Universal Database" (DB2 UDB), que puede almacenar y hacer búsquedas no solamente de datos alfanuméricos sino también de imágenes, audio, video y otros objetos [IBM2005].

DB2 es completamente escalable, veloz y confiable. Corre en modo nativo en casi todas las plataformas, como Windows NT (R), Sun Solaris, HP-UX, AIX(R), OS/400 y OS/2(R). Permite que los clientes manejen, integren y analicen información de la más amplia variedad de fuentes. Software de la IBM en base de datos con el soporte más amplio para estándares abiertos.

DB2 no es un producto nuevo, fue construido en base a dos productos incluidos en el DB2 de AIX en el año 1994: DB2 Common Server, que para propósitos generales incluía funciones avanzadas para el mercado de servidores de bases de datos, con soporte de hardware SMP y OLTP; y el DB2 Parallel Edition, que fue desarrollado para soportar aplicaciones de gran escala, como Data Warehousing y Data Mining [IBM2005].

2.2. CARACTERÍSTICAS DE DB2.

DB2 es el producto principal de la estrategia de Data Management de IBM. DB2 es un sistema para administración de bases de datos relacionales (RDBMS) multiplataforma, especialmente diseñada para ambientes distribuidos, permitiendo que los usuarios locales compartan información con los recursos centrales. A continuación se detallan algunas de las características de este producto [IBM2005]:

- DB2 incluye características de integridad, asegurando la protección de sus datos aún en caso de que los sistemas sufran un colapso; y de seguridad, permitiendo realizar respaldos en línea con distintos grados de granularidad, sin que esto afecte la disponibilidad de acceso a los datos por parte de los usuarios.

- Provee la capacidad de hacer frente a múltiples necesidades, desde procesamiento transaccional de misión crítica (OLTP), hasta análisis en línea exhaustivo de los datos para el soporte a la toma de decisiones (OLAP).

- Sus características distintivas de escalabilidad le permiten almacenar información en un amplio rango de equipos, desde una PC portátil hasta un complejo ambiente de mainframes procesando en paralelo.

- Incluye tecnología basada en Web que permite generar aplicaciones en sus Intranets y responder a las oportunidades de negocios disponibles en Internet, además DB2 provee soporte a Java.

- La primera versión de DB2 para NT fue reconocida en el mercado como una base de datos muy poderosa, pero difícil de instalar y usar. En esta versión (DB2), IBM agregó muchas herramientas gráficas para facilitar el uso tanto de usuarios, como administradores y desarrolladores. Incluye guías para operaciones como instalación, configuración de performance, setup, etc. Además, se agregaron herramientas para facilitar las tareas de integración con otras bases de datos, tecnologías de networking y desarrollo de aplicaciones.

- DB2 UDB es, además, multiplataforma (16 plataformas - 10 no IBM), brinda soporte a un amplio rango de clientes, soporta el acceso de los datos desde Internet y permite almacenar todo tipo de datos incluyendo texto, audio, imágenes y video o cualquier otro definido por el usuario.

- Las herramientas de conectividad permiten acceder a los datos más allá de donde ellos se encuentren. El slogan 'cualquier cliente, a cualquier servidor, en cualquier red' está

completamente sustentado por la funcionalidad que sus herramientas ofrecen. EL DB2 Connect le permiten acceder a sus datos de DB2 en mainframe o AS/400, desde Windows NT, Windows 95 / 98, OS/2 o cualquiera de los Unix soportados. Además, el producto Datajoiner posibilita acceder de forma única y transparente a los datos residentes en Oracle, Sybase, Informix, Microsoft SQL Server, IMS, VSAM y otros.

- DB2 provee la infraestructura necesaria para soportar el proceso de toma de decisiones en cualquier tamaño y tipo de organización. Es el producto dirigido a resolver la problemática a nivel departamental (Data Marts), ya que un único producto provee la capacidad para acceder a datos en Oracle, Sybase, Informix, Microsoft SQL Server, VSAM o IMS, además de la familia DB2. Permite de forma totalmente gráfica acceder, transformar y distribuir los datos automáticamente y sin programar una línea de código.

- DB2 posibilita el análisis orientado al descubrimiento de información escondida en los datos, realizando modelización predictiva, segmentación de la base de datos, análisis de vínculos, o detección de desviaciones. Incluye las siguientes técnicas: clustering (segmentación), clasificación, predicción, descubrimiento asociativo, descubrimiento secuencial de patrones y secuencias temporales. Todas las técnicas mencionadas permiten realizar segmentación de clientes, detección de fraudes, retención de clientes, ventas cruzadas, etc.

2.3. ALGUNAS RAZONES PARA ELEGIR DB2 [IBM2005].

2.3.1. DB2 es de Fácil Manejo.

Muchos expertos de la industria y usuarios han elogiado las nuevas herramientas que IBM desarrolló para facilitar la administración y uso del DB2. Utiliza una interfase gráfica, estilo browser, para acceder y manejar objetos de la base de datos. Incluye “smart.guides” que facilitan la tarea de configuración, guiándolo paso a paso para lograr un rendimiento óptimo de la base de datos y para asistir al usuario en la creación de teclas con plantillas predefinidas.

2.3.2. Por las Aplicaciones existentes para DB2.

Una cantidad significativa de empresas importantes del mundo confían en DB2 para manejar aplicaciones críticas. Se pueden poner en funcionamiento todas estas aplicaciones de negocios:

- On-line Transaction Processing (OLTP)
- Data Warehousing
- Decision Support
- Data Mining

Además se deben considerar los beneficios a corto y largo plazo al utilizar un código base unificado para todas sus aplicaciones, y para todas sus plataformas, desde laptops hasta sistemas de procesamiento paralelo.

Se dispone de numerosas aplicaciones desarrolladas a nivel mundial, entre ellas las de SAP, People Soft y Baan.

2.3.3. DB2 incluye herramientas de replicación.

La replicación de datos es la tecnología clave para aprovechar todo el poder de los ambientes distribuidos ya que le permite enviar los datos a cualquier sitio para cubrir todos los requerimientos de su empresa; desde oficinas centrales a sucursales, usuarios móviles, proveedores, clientes y socios de negocios.

DB2 incluye todo lo necesario para implementar una solución de replicación de datos en cualquier tipo de ambiente distribuido o heterogéneo.

2.3.4. BB2 brinda Soporte OLAP.

DB2 ofrece nuevas capacidades para ayudarlo a realizar análisis multidimensional y procesamiento analítico en línea (OLAP). Incluye funciones de ROLLUP, CUBE y grouping sets. Soporta STAR JOINS.

2.3.5. DB2 está lista para Internet.

Gracias a su alcance global y bajo costo, Internet puede ser una solución de negocios muy poderosa. Si se quiere realmente aprovechar las oportunidades de negocios utilizando la Web, la base de datos que seleccione debe estar preparada para brindar acceso rápido y alta disponibilidad a información que requiere actualización constante. DB2 provee la capacidad de hacer backups en línea, evitando interrupciones en la disponibilidad del sistema. Permite realizar operaciones comerciales garantizando un alto nivel de seguridad y confiabilidad.

2.4. ALMACÉN DE DATOS (DATA WAREHOUSE).

Los Data Warehousing, nacen debido a la necesidad de contar con información de apoyo a la toma de decisiones, dado que los datos operacionales no cumplen eficientemente con este objetivo. Esto conlleva a desarrollar Data Warehouses para poner el uso estratégico de la información como generador de ventajas competitivas.

Hay diferentes definiciones de Data Warehouse según el autor que se consulte, la más utilizada por su sencillez, claridad y fácil comprensión es la entregada por Inmon, considerado el padre del Data Warehouse;

"Un Data Warehouse es una colección de datos integrados, temáticos, no volátiles y variantes en el tiempo, organizados para soportar necesidades empresariales orientadas a la toma de decisiones" [INM2002].

Se puede concluir, que un Data Warehouse, es el proceso de extraer y filtrar datos de las operaciones comunes de las empresas, procedentes de los distintos subsistemas operacionales, para transformarlos, integrarlos, sumariarlos y almacenarlos en un depósito o almacén de datos, para poder acceder a ellos cada vez que se necesite mediante mecanismos flexibles para el usuario, en la figura Nº 2.1 se presenta la Arquitectura base de un Sistema de Data Warehousing.

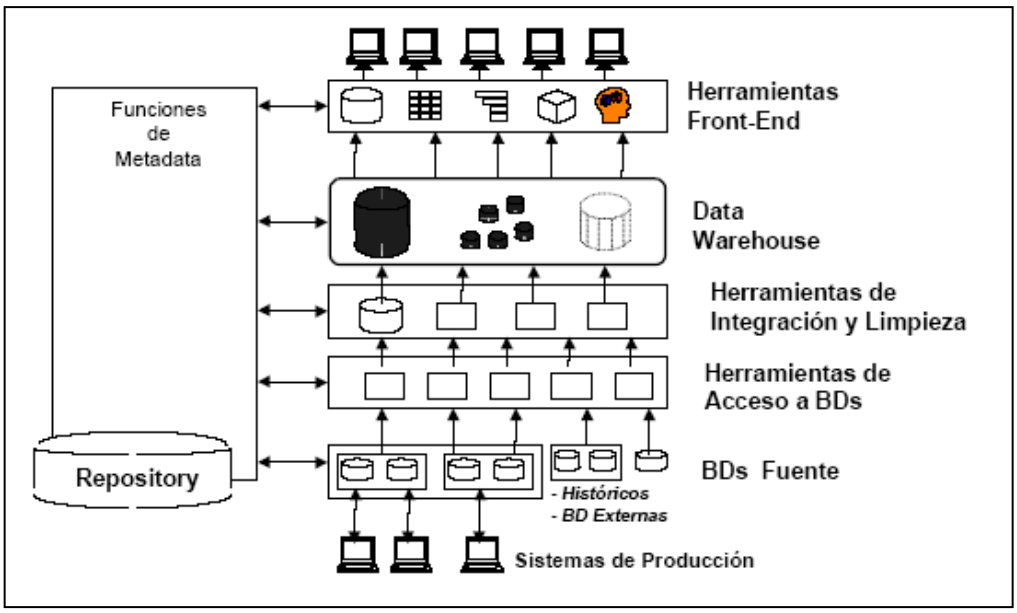


Figura Nº 2.1 "Arquitectura base de un Sistema de Data Warehousing".

2.4.1. Características de los Sistemas de Data Warehousing.

Las características de un Data Warehouse según Inmon, son [INM2002]:

2.4.1.1. Integrado.

El aspecto más importante en los Data Warehouses es que la información encontrada siempre está integrada. La integración de los datos se muestra de muchas maneras: en conversiones de nombres consistentes, en la medida uniforme de las variables, en la conversión de estructuras consistentes, en atributos físicos de los datos consistentes, fuentes múltiples, entre otros.

Los datos almacenados en el Data Warehouse, deben integrarse en una estructura consistente, por lo que las inconsistencias existentes en 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.

2.4.1.2. Temáticos.

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.

2.4.1.3. No Volátil.

La información es útil solo cuando es estable. Los datos operacionales cambian sobre una base momento a momento. La perspectiva más grande, esencial para el análisis y la toma de decisiones, requiere una base de datos estable.

Es importante mencionar que el período de tiempo cubierto por un Data Warehouse varía entre 2 y 10 años. [HER2003]

2.4.1.4. Variante en el Tiempo.

El tiempo es la parte implícita de la información en un Data Warehouse. En los sistemas operacionales, los datos siempre reflejan el estado de la actividad del negocio en el momento presente. Por el contrario, la información contenida en el Data Warehouse sirve, entre otras cosas, para realizar análisis de tendencias. Por lo tanto, el Data Warehouse se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.

2.4.2. Valor del DW para la Toma de Decisiones.

El valor de un DW queda descrito en tres dimensiones [INM1994]:

- **Mejorar la Entrega de Información:** información completa, correcta, consistente, oportuna y accesible. Información que la gente necesita, en el tiempo que la necesita y en el formato que la necesita.
- **Facilitar el Proceso de Toma de Decisiones:** con un mayor soporte de información se obtienen decisiones más rápidas; así también, la gente de negocios adquiere mayor confianza en sus propias decisiones y las del resto, y logra un mayor entendimiento de los impactos de sus decisiones.

- **Impacto Positivo sobre los Procesos Empresariales:** cuando a la gente accede a una mejor calidad de información, la empresa puede:
    - ✓ Eliminar los retardos de los procesos empresariales que resultan de información incorrecta, inconsistente y/o no existente.
    - ✓ Integrar y optimizar procesos empresariales a través del uso compartido e integrado de las fuentes de información.
    - ✓ Eliminar la producción y el procesamiento de datos que no son usados ni necesarios, producto de aplicaciones mal diseñados o ya no utilizadas.

2.5. OLAP.

Un análisis acertado de los datos conlleva una toma de decisiones de negocio correcta, lo que se traduce en un aumento de los beneficios. En esta línea, una de las técnicas mas utilizadas a la hora de dar soporte a la toma de decisiones empresariales. Es el OLAP: Proceso Analítico en Línea que permite al usuario seleccionar y extraer la información desde diferentes puntos de vista, y que se traduce en un conjunto de técnicas y unos estándares definidos por organismos de estandarización (OLAP Council, Analytical Solutions Forum,...). Con OLAP, un usuario podría analizar sus resultados de las ventas e ir profundizando en detalle en cada región, canal de venta, equipos de ventas o vendedores hasta encontrar la información relevante para su toma de decisiones. Este es un sencillo ejemplo de cómo OLAP permite hacer un análisis correcto y rápido de la información desde los puntos de vista o "dimensiones" que el usuario desee.

Estos retos son enfrentados con herramientas avanzadas de peticiones (*queries*), las cuáles ocultan al usuario final la complejidad de las base de datos. Ésta es la función de las herramientas OLAP. Todas las organizaciones tienen datos multidimensionales y la complejidad no es necesariamente una función del tamaño de la compañía. Aún a las más pequeñas compañías les gustaría poder rastrear sus ventas por producto, vendedor, geografía, cliente y tiempo. Las organizaciones han buscado durante mucho tiempo herramientas para acceder, navegar y analizar información multidimensional de una manera fácil y natural.

2.5.1. Utilidad del análisis OLAP en los negocios.

Con la implementación del análisis OLAP las empresas pueden reducir costos externos en IT (hardware, soporte, consultoría o licenciamiento de software), ahorrar costos en departamentos administrativos, aumentar ingresos a través de un mejor análisis de mercadeo y ventas, ahorrar costos no relacionados en IT (inventarios, pérdidas, financiación), mejorar la satisfacción del cliente a través del mejoramiento de la calidad de los productos o servicios, mejorar las decisiones de negocio a través de un minucioso y oportuno análisis y realizar reportes más rápidos y exactos.

2.5.2. Modelamiento o Esquema Multidimensional.

Es un nuevo modelo de datos que sirve de base para las aplicaciones del tipo procesamiento analítico en línea, se usa para conceptualizar modelos en los negocios como un conjunto de medidas descriptivas, especialmente útil para resumir y ordenar datos para el análisis [JGO1998]. Además este tipo de modelamiento de datos no está ligado a la implementación física de ningún tipo de base de datos, ya que es un modelo de diseño lógico que busca representar datos en un esquema estándar, con alto rendimiento y disposición de los datos de tal forma que permita el almacenamiento, recuperación eficiente y conveniente de grandes volúmenes de datos íntimamente relacionados y almacenados, vistos y analizados de diferentes perspectivas. El modelamiento multidimensional busca ver la información de negocios en forma integrada en un momento en el tiempo.

La estructura básica para un modelo multidimensional está definida por dos elementos: esquemas y tablas. Como en cualquier base de datos relacional, este tipo de base de datos se compone de tablas, en el modelo multidimensional hay dos tipos básicos de tablas: las Fact, que contienen valores de las medidas de negocios y las Dimensionales, que contienen el detalle de los valores que se encuentran asociados a las tablas Fact.

Los esquemas corresponden a la colección de tablas que conforman el modelo, entre otros se encuentra el Esquema Estrella y el Esquema Copo de Nieve.

2.5.2.1. Esquema Estrella.

Su estructura básica es una tabla central y un conjunto de tablas que la atienden radialmente, como se ve en la figura Nº 2.2. Este esquema deriva su nombre del hecho que su diagrama forma una estrella, con puntos radiales desde el centro. El centro de la estrella consiste de una o más tablas Fact y las puntas de la estrella son las tablas dimensionales.

Este modelo es asimétrico, pues hay una tabla dominante en el centro con varias conexiones a la tabla Fact y ninguna más.

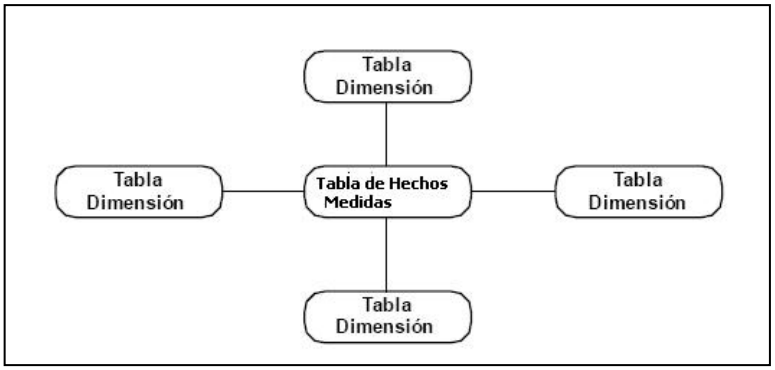


Figura Nº 2.2 "Estructura del esquema Estrella".

2.5.2.2. Esquema Copo de Nieve.

La diferencia de este esquema con el anterior, es la estructura de las tablas dimensionales que en el modelo copo de nieve están normalizadas. Cada tabla dimensional contiene sólo el nivel que es clave primaria en la tabla y la foreign key de su parentesco del nivel más cercano del diagrama. En la figura Nº 2.3 se puede ver este esquema.

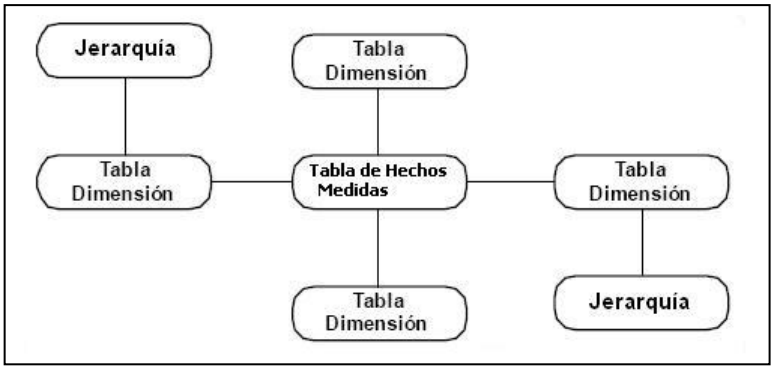


Figura Nº 2.3 "Estructura del esquema Copo de Nieve".

2.5.3. Representación de un Cubo.

En un modelo de datos OLAP, la información es vista como cubos, los cuáles consisten de categorías descriptivas (dimensiones) y valores cuantitativos (medidas).

El modelo de datos multidimensional simplifica a los usuarios el formular peticiones complejas, arreglar datos en un reporte, cambiar de datos de resumen a datos de detalle y filtrar o seccionar los datos en subconjuntos significativos.

Por ejemplo, las dimensiones típicas de un cubo que contenga información de ventas, incluiría tiempo, geografía, producto, canal, organización y escenario (planeado o real). Las medidas típicas incluirían ventas en dólares (u otra moneda), unidades vendidas, número de personas, ingresos y gastos. La figura Nº 2.4 muestra un cubo con las dimensiones producto, fecha y país.

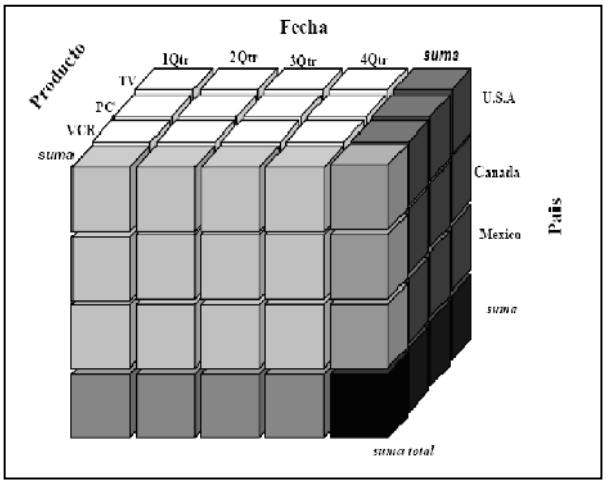


Figura Nº 2.4 "Ejemplo de Cubo".

Dentro de cada dimensión de un modelo de datos OLAP, los datos se pueden organizar en una jerarquía que represente niveles de detalle de los datos. Por ejemplo, dentro de la dimensión de tiempo, se puede tener estos niveles: años, meses y días; de manera similar, dentro de la dimensión geografía, Se puede tener estos niveles: país, región, estado/provincia y ciudad. Una instancia particular del modelo de datos OLAP tendrá valores para cada nivel en la jerarquía. Un usuario que vea datos OLAP se moverá entre estos niveles para ver información con mayor o menor detalle.

2.5.4. Implementación del OLAP.

Los cubos, las dimensiones y las jerarquías son la esencia de la navegación multidimensional del OLAP. Al describir y representar la información en esta forma, los usuarios pueden navegar intuitivamente en un conjunto complejo de datos. Sin embargo, el solo describir el modelo de datos en una forma más intuitiva, hace muy poco para ayudar a entregar la información al usuario más rápidamente.

Un principio clave del OLAP es que los usuarios deberían de ver tiempos de respuesta consistentes para cada vista de datos que requieran. Dado que la información se colecta en el nivel de detalle solamente, el resumen de la información es usualmente calculado por adelantado. Estos valores precalculados, son la base de las ganancias de desempeño del OLAP.

En los primeros días de la tecnología OLAP, la mayoría de las compañías asumía que la única solución para una aplicación OLAP era un modelo de almacenamiento no relacional. Después, otras compañías descubrieron que a través del uso de estructuras de base de datos (esquemas de estrella y de copo de nieve), índices y el almacenamiento de agregados, se podrían utilizar sistemas de administración de bases de datos relacionales (*RDBMS*) para el OLAP.

Estos vendedores llamaron a esta tecnología OLAP Relacional (ROLAP). Las primeras compañías adoptaron entonces el término OLAP Multidimensional (MOLAP), estos conceptos, MOLAP y ROLAP, se explican con más detalle en los siguientes párrafos. Las implementaciones MOLAP normalmente se desempeñan mejor que la tecnología ROLAP, pero tienen problemas de escalabilidad. Por otro lado, las implementaciones ROLAP son más escalables y son frecuentemente atractivas a los clientes debido a que aprovechan las inversiones en tecnologías de bases de datos relacionales preexistentes. Un desarrollo reciente ha sido la solución OLAP híbrida (HOLAP), la cual combina las arquitecturas ROLAP y MOLAP para brindar una solución con las mejores características de ambas: desempeño superior y gran escalabilidad. Un tipo de HOLAP mantiene los registros de detalle (los volúmenes más grandes) en la base de datos relacional, mientras que mantiene las agregaciones en un almacén MOLAP separado.

2.5.4.1. Sistemas MOLAP (OLAP Multidimensional).

La arquitectura MOLAP usa bases de datos multidimensionales para proporcionar el análisis, su principal premisa es que el OLAP está mejor implantado almacenando los datos multidimencionalmente. Por el contrario, la arquitectura ROLAP cree que las capacidades OLAP están perfectamente implantadas sobre bases de datos relacionales.

Un sistema MOLAP usa una base de datos propietaria multidimensional, en la que la información se almacena multidimensionalmente, para ser visualizada en varias dimensiones de análisis.

El sistema MOLAP utiliza una arquitectura de dos niveles [SAS2003]: las bases de datos multidimensionales y el motor analítico. La base de datos multidimensional es la encargada del manejo, acceso y obtención del dato.

El nivel de aplicación es el responsable de la ejecución de los requerimientos OLAP. El nivel de presentación se integra con el de aplicación y proporciona un interfaz a través del cual los usuarios finales visualizan los análisis OLAP. Una arquitectura cliente/servidor permite a varios usuarios acceder a la misma base de datos multidimensional.

La información procedente de los sistemas operacionales, se carga en el sistema MOLAP, mediante una serie de rutinas por lotes. Una vez cargado el dato elemental en la Base de Datos multidimensional (MDDB), se realizan una serie de cálculos por lotes, para calcular los datos agregados, a través de las dimensiones de negocio, rellenando la estructura MDDB.

Tras rellenar esta estructura, se generan unos índices y algoritmos de tablas *hash* para mejorar los tiempos de accesos a las consultas. Una vez que el proceso de compilación se ha acabado, la MDDB está lista para su uso. Los usuarios solicitan informes a través de la interfase, y la lógica de aplicación de la MDDB obtiene el dato.

La arquitectura MOLAP requiere unos cálculos intensivos de compilación. Lee de datos precompilados, y tiene capacidades limitadas de crear agregaciones dinámicamente o de hallar ratios que no se hayan precalculados y almacenados previamente.

2.5.4.2. Sistemas ROLAP (OLAP Relacional).

La arquitectura ROLAP, accede a los datos almacenados en un Data Warehouse para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales.

El sistema ROLAP utiliza una arquitectura de tres niveles [SAS2003]. La base de datos relacional maneja los requerimientos de almacenamiento de datos, y el motor ROLAP proporciona la funcionalidad analítica. El nivel de base de datos usa bases de datos relacionales para el manejo, acceso y obtención del dato. El nivel de aplicación es el motor que ejecuta las consultas multidimensionales de los usuarios.

El motor ROLAP se integra con niveles de presentación, a través de los cuáles los usuarios realizan los análisis OLAP. Después de que el modelo de datos para el Data Warehouse se ha definido, los datos se cargan desde el sistema operacional. Se ejecutan rutinas de bases de datos para agregar el dato, si así es requerido por el modelo de datos. Se crean entonces los índices para optimizar los tiempos de acceso a las consultas.

Los usuarios finales ejecutan sus análisis multidimensionales, a través del motor ROLAP, que transforma dinámicamente sus consultas a consultas SQL. Se ejecutan estas consultas SQL en las bases de datos relacionales, y sus resultados se relacionan mediante tablas cruzadas y conjuntos multidimensionales para devolver los resultados a los usuarios.

La arquitectura ROLAP es capaz de usar datos precalculados si éstos están disponibles, o de generar dinámicamente los resultados desde los datos elementales si es preciso. Esta arquitectura accede directamente a los datos del Data Warehouse, y soporta técnicas de optimización de accesos para acelerar las consultas. Estas optimizaciones son, entre otras,

particionado de los datos a nivel de aplicación, soporte a la desnormalización y *joins* múltiples.

2.5.5. Operaciones de Usuario.

La funcionalidad de los sistemas OLAP se caracteriza por ser un análisis multidimensional de datos corporativos, que soportan los análisis del usuario y unas posibilidades de navegación, seleccionando la información a obtener.

Normalmente este tipo de selecciones se ve reflejada en la visualización de la estructura multidimensional, en unos campos de selección que permitan elegir el nivel de agregación (jerarquía) de la dimensión, y/o la elección de un dato en concreto, la visualización de los atributos del sujeto, frente a una(s) dimensiones en modo tabla, pudiendo con ello realizar, entre otras las siguientes acciones [LFD2003]:

- **Slice:** Esta operación consiste en reducir la dimensionalidad del esquema eliminando alguna dimensión. Aplicar esta operación implica pérdida de detalle en los hechos (aumento en la granularidad) por lo que tendremos que usar operadores de agregación para la obtención de los nuevos, en la figura Nº 2.5 se puede ver su representación gráfica.

  Definición Algebraica:

$$\prod\left(C_s^n, D_1,....,D_m,D_{m+1}= d_{m+1},....,D_n = d_n\right)= C_s^{'m} \qquad \text{m <= n}$$

  Sea C_s^n el cubo de n dimensiones y de nombre s.

  D las dimensiones.

  d las condiciones y dimensiones resultantes.

  m el número de dimensiones del cubo resultante.

Ejemplo:

$$\prod\left(C_c^3, año, tienda, producto = "Pc"\right) and \prod\left(C_c^3, tienda, producto = "Pc", año = "1996"\right)$$

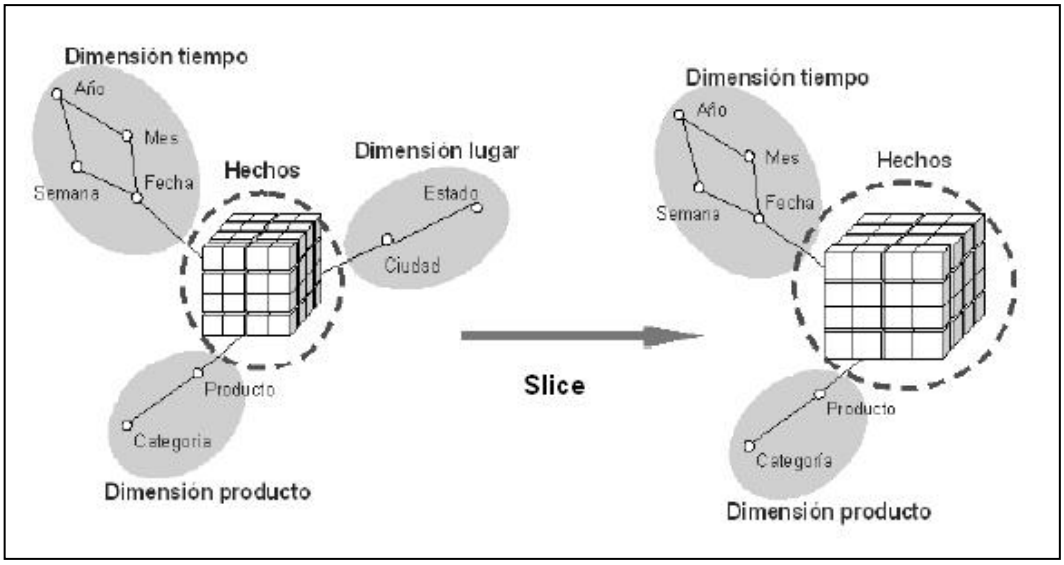


Figura Nº 2.5 "Ejemplo Operación Slice".

- **Dice:** Esta operación lo que hace es restringir los valores que consideramos en las dimensiones según alguna condición. No se modifica la estructura del cubo en cuanto a las dimensiones y niveles de éstas, sino restringiendo los valores que se consideren. En este caso se modifica el nivel de detalle de los hechos pero si su número, dado que aquellos que tuvieran como coordenadas valores que no se consideraron se deben eliminar.

Definición Algebraica:

$$\sigma\left(C_s^m, P\right) = C_s^{'m}$$

m número de dimensiones del cubo C de nombre s.
p condición para la dimensión.

Ejemplo:

$$\sigma\left(C_c^3, año = "1996" o raño = "1997"\right)$$

- **Drill-up:** Significa subir en la jerarquía, esto es aumentar el tamaño del grano al que están definidos los hechos. Al aplicar esta operación se necesita resumir información para adaptar el nivel de detalle de los hechos, en este proceso de resumen se utilizarán operadores de agregación, en la figura Nº 2.6 se puede visualizar gráficamente esta operación.

  Definición Algebraica:

$$\Uparrow \left(C_s^n, D_1, ..., D_{n-1}\right) = C_{s'}^{'n-1}$$

  n número de dimensiones del cubo C de nombre s.

  v medidas del hecho.

- **Drill-down:** Esta operación es lo contrario de drill-up, ahora lo que se pretende es reducir el nivel de grano obteniendo un mayor nivel de detalle. Esto se traduce en cambiar el nivel de definición de los hechos a niveles inferiores de las jerarquias, en la figura Nº 2.6 se puede ver un ejemplo gráfico de esta operación.

  Definición Algebraica:

$$\Downarrow \left(C_s^n, C_{n+1}\right) = C_{s'}^{'n+1}$$

  n número de dimensiones del cubo C de nombre s.

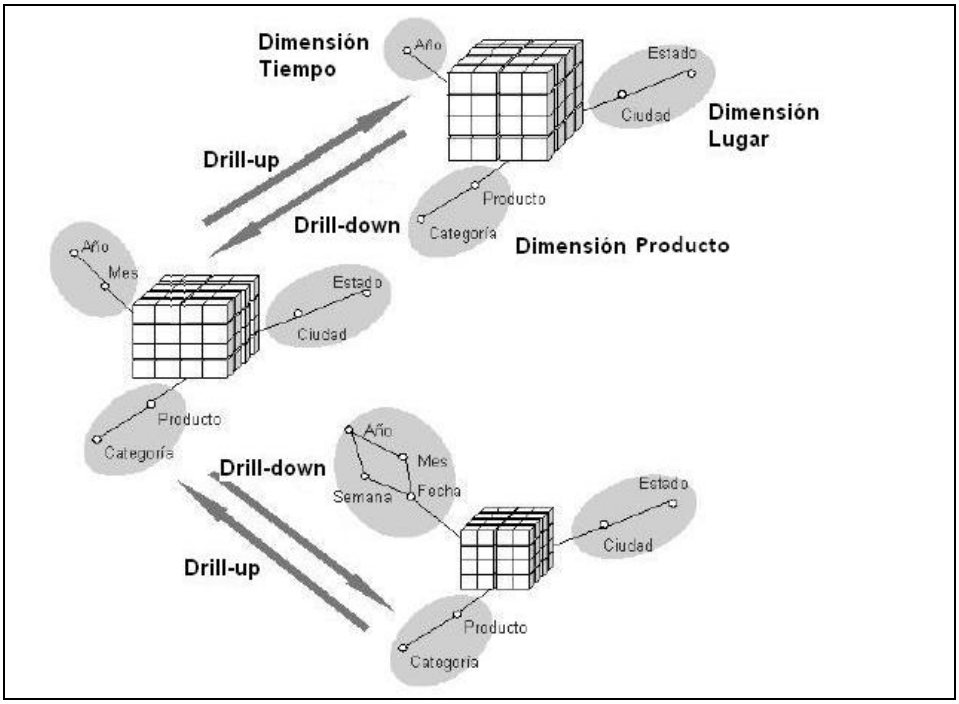


Figura Nº 2.6 "Ejemplo Operaciones Drill-up, Drill-down".

- **Rotación:** Esta operación lo que persigue es la modificación de la definición de la estructura del cubo. Lo que implica es un cambio en el orden de las dimensiones, en la figura 2.7 se ve un ejemplo gráfico de la operación.

  Definición Algebraica:

  $$\partial\left(C_s^n, D_1, ..., D_n\right) = C_{s'}^{'n}$$

  Donde $C_{s'}^{'n}$ es un cubo con las mismas dimensiones del cubo C_s^n en distinto orden.

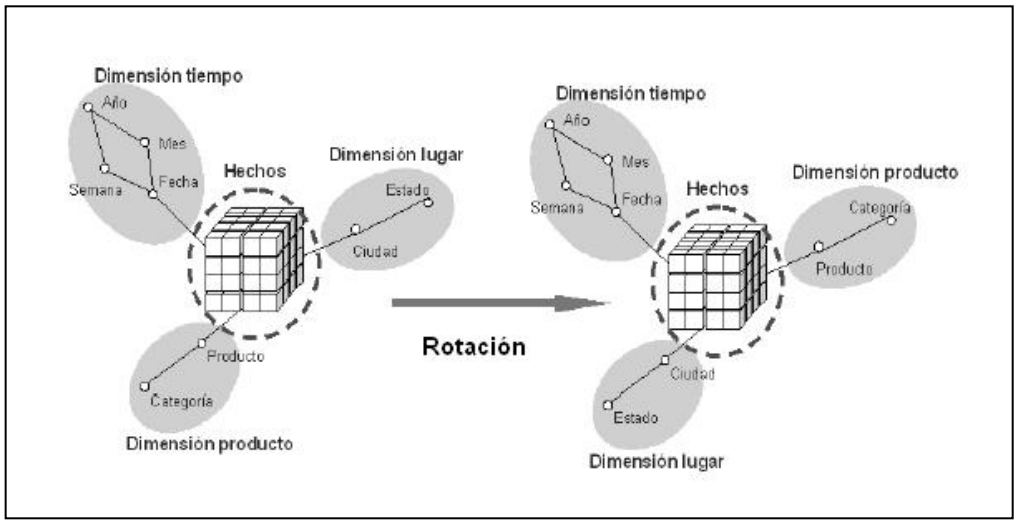


Figura Nº 2.7 "Ejemplo Operación Rotación".

Capítulo 3

“Operadores Cube y Rollup”

3. OPERADORES CUBE Y ROLLUP.

El cubo de datos es una estructura de datos especial usada en los Data Warehousing y OLAP para representar el modelo multidimensional de datos. En el presente capítulo se describe el operador CUBE, operador que simula un cubo multidimensional y que permite que estos se puedan consultar con instrucciones breves en SQL. Los temas tratados en este capítulo hacen referencia a [GRA1996] y son la base para la implementación realizada en el capítulo 4.

3.1. LA PROBLEMÁTICA DEL GROUP BY.

El operador relacional GROUP BY divide una tabla en grupos. Cada grupo se agrega con una función. La función de agregación resume algún grupo de columnas retornando el valor de cada grupo, como se ve en la figura Nº 3.1.

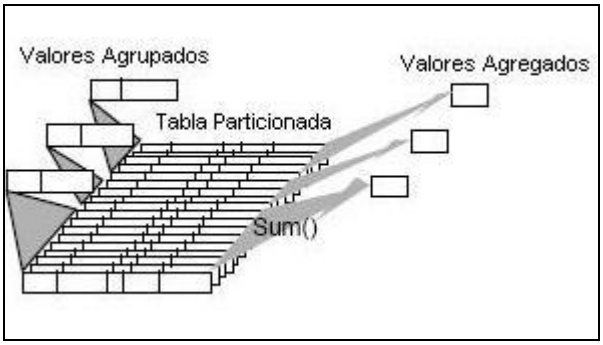


Figura Nº 3.1 "Operador Group By".

Ciertas formas de análisis de datos son difíciles y algunas casi imposibles con los constructores SQL. Tres problemas comunes son:

- Histogramas.
- Total de Roll-up y sub-total de drill-down.
- Tablas Cruzadas.

El primer problema es que el GROUP BY no permite agregación directa sobre las categorías analizadas. Sería fácil hacer un histograma sobre la tabla tiempo (Representada con los atributos mostrados en la tabla Nº 3.1), si los valores de la función Group By son permitidos, por ejemplo, se podría consultar la temperatura diaria máxima registrada con la sentencia SQL de la figura Nº 3.2.

Tiempo(UCT)	Latitud	Longitud	Altitud (m)	Temp. ºC	Presión (mb)
27/11/94:1500	37:58:33N	122:45:28W	102	21	1009
27/11/94:1500	34:16:18N	27:05:55W	10	23	1024

Tabla Nº 3.1 "Tabla tiempo".

```
SELECT      día, nación, MAX(Temp)
FROM        tiempo
GROUP BY    Día(Time) AS día, País(Latitud, Longitud) AS nación;
```

Figura Nº 3.2 "SQL temperatura máxima registrada".

Sin embargo, algunos sistemas SQL soportan histogramas (historiales), pero el SQL estándar no lo hace. Para solucionar este problema se debe construir una tabla valorada y luego agregarla sobre la tabla resultante (eso quiere decir, que se debe crear una tabla temporal y desde esa tabla sacar los datos que se quieren). La sentencia en SQL92 de la figura Nº 3.3 muestra como se hace.

```
SELECT          día, nación, MAX(Temp)
FROM  (SELECT día (Time) AS día,
                País (Latitud,Longitud) AS nación, Temp
                FROM tiempo ) AS foo
GROUP BY día, nación;
```

Figura Nº 3.3 "Creación de un Histograma".

El segundo problema es para la agregación del Roll-up o informes Drill Down en donde se tiene que almacenar cada nivel, por ejemplo el subtotal de la agregación. Los reportes generalmente agregan datos en un nivel alto, luego, sucesivamente, a los niveles más bajos. El informe sobre venta de autos representado en la tabla Nº 3.2 muestra la idea de este punto, esta tabla no es relacional, los valores nulos en la llave primaria no están permitidos. Eso tampoco es conveniente, el número de columnas aumenta como la potencia del número de atributos agregados. Los datos se agregan por modelo, luego por año y luego por color. El informe muestra datos agregados en tres niveles. Yendo a la parte superior de los niveles se llama Drill-up, yendo a la parte inferior se llama Drill-down.

Roll Up de las Ventas por Modelo por Año por Color					
Modelo	**Año**	**Color**	**Ventas por Modelo por Año por Color**	**Ventas por Modelo por Año**	**Ventas por Modelo**
Chevy	1994	Negro	50		
		Blanco	40		
				90	
	1995	Negro	85		
		Blanco	115		
				200	
					290

Tabla Nº 3.2 "Informe Venta de Autos (No relacional)".

La idea de la solución es agregar columnas nuevas para cada combinación de atributos agregados en los cuales se está interesado; solución pobre, redundante, perdida de almacenaje, e introducir el valor "Todos". Este valor representa el conjunto de todos los valores que existen en cierta columna. Un ejemplo de esta solución se ve en la sentencia SQL estándar de la figura Nº 3.4, que construye el "Resumen de Ventas de Autos" de la tabla Nº 3.3, esta es una tabla relacional y una conveniente representación donde el valor simulado "Todos" se añadió para completar los campos súper agregados.

```
SELECT Modelo, Todos, Todos, SUM(Ventas)
FROM Ventas
WHERE Modelo = 'Chevy'
GROUP BY Modelo
UNION
SELECT Modelo, Año, Todos, SUM(Ventas)
FROM Ventas
WHERE Modelo = 'Chevy'
GROUP BY Model, Year
UNION
SELECT Modelo, Año, Color, SUM(Ventas)
FROM Ventas
WHERE Modelo = 'Chevy'
GROUP BY Modelo, Año, Color;
```

Figura Nº 3.4 "Sentencia SQL Resumen de ventas".

Resumen de Ventas			
Modelo	**Año**	**Color**	**Unidades**
Chevy	1994	Negro	50
Chevy	1994	Blanco	40
Chevy	1994	Todos	90
Chevy	1995	Negro	85
Chevy	1995	Blanco	115
Chevy	1995	Todos	200
Chevy	Todos	Todos	290

Tabla Nº 3.3 "Informe Venta de Autos (Relacional)".

El tercer problema, construir una tabla cruzada con SQL es aún más desalentador puesto que el resultado no es realmente un objeto emparentado como se puede ver en los datos de la tabla Nº 3.4, la fila inferior y la columna derecha son "inusuales". Esta tabla cruzada es una agregación de dos dimensiones.

Tabla Cruzada Ventas de Chevy			
Chevy	**1994**	**1995**	**Total** (Todos)
Negro	50	85	135
Blanco	40	115	155
Total (Todos)	90	200	290

Tabla Nº 3.4 "Tabla cruzada Ventas Chevy".

Si se agregan otros modelos del automóvil, se convierte en una agregación 3D. Por ejemplo, los datos para los productos de Ford agregan un plano adicional a la tabla cruzada, como muestra la tabla Nº 3.5.

Tabla Cruzada Ventas de Ford			
Ford	1994	1995	**Total** (Todos)
Negro	50	85	135
Blanco	10	75	85
Total (Todos)	60	160	220

Tabla Nº 3.5 "Tabla cruzada ventas Ford".

La representación del arreglo de tablas cruzadas es equivalente a la representación relacional usando el valor "Todos". Ambos generalizan a tablas cruzadas de N-dimensiones. La representación y el uso de uniones del GROUP BY "solucionan" el problema, éstas representan datos agregados en un modelo de datos relacional.

3.2. CUBE Y ROLLUP.

El operador CUBE construye una tabla con todos los valores agregados y es un operador relacional. Su definición es:

v1, v2,..., vn, f()
v1, v2,..., 'Todos', f()
.
.
v1, 'Todos',..., 'Todos', f()
'Todos', 'Todos',..., 'Todos', f()……., donde f() es una función de agregación.

3.2.1 El Operador de Datos CUBE.

El operador Cube es la generalización en N-dimensiones de funciones de agregación simples. El cubo de datos de 0D es un punto, el de 1D es una línea con un punto, el de 2D es una tabla cruzada, un plano, dos líneas y un punto y el de 3D es un cubo con tres intersecciones de tablas cruzadas en 2D, como se representa en la figura Nº 3.5.

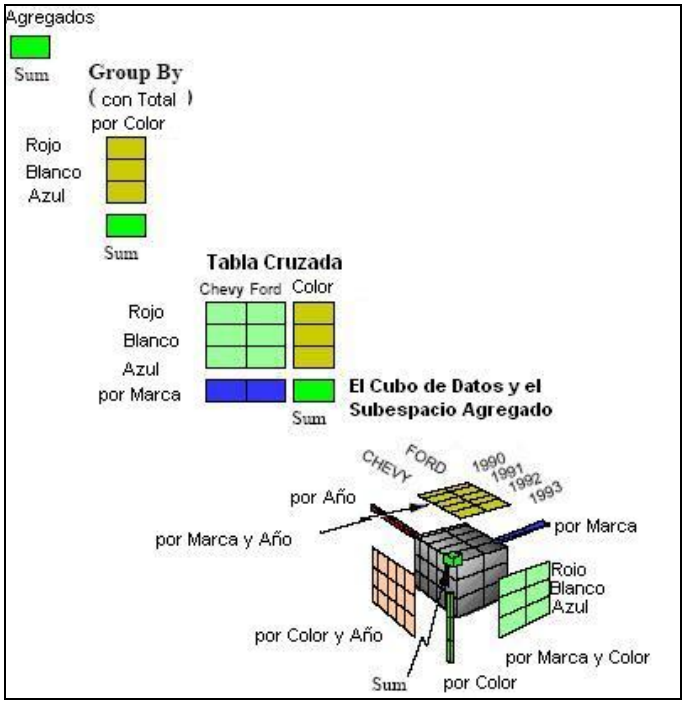


Figura Nº 3.5 "Operador Cube".

3.2.1.1. Sintaxis del Operador CUBE.

En la figura Nº 3.6 se ve un ejemplo básico de sintaxis del operador CUBE.

```
SELECT      día, nación, MAX(Temp)
FROM        tiempo
GROUP BY    CUBE ( Día (Time) AS dia, Pais(Latitud, Longitud) AS
Nacion);
```

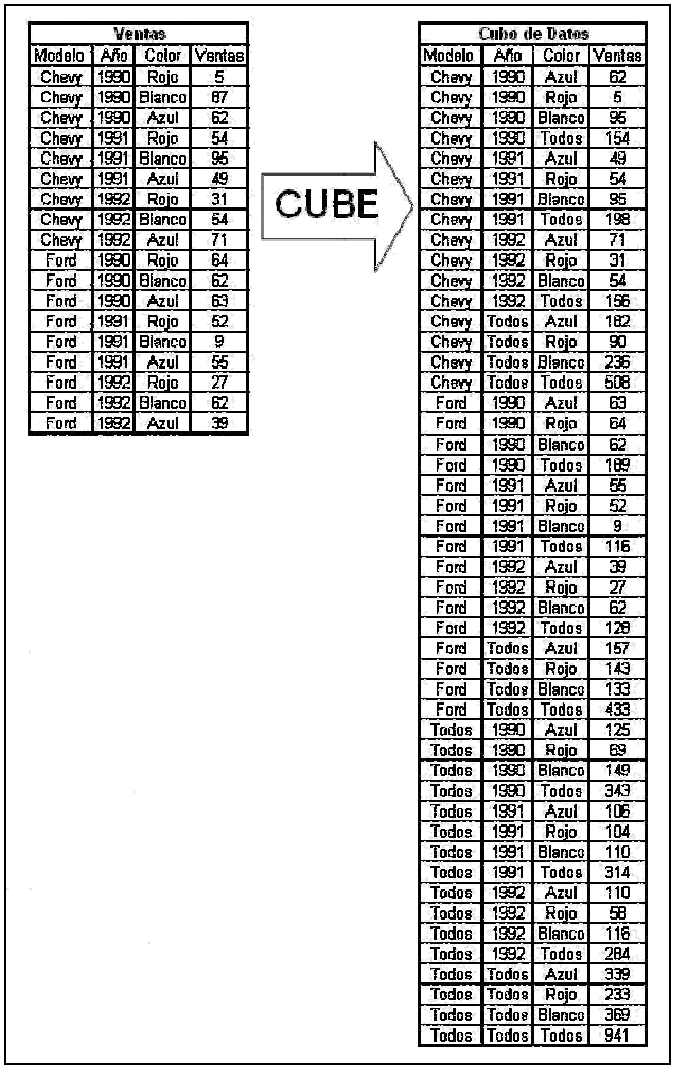
Figura Nº 3.6 "Ejemplo Sintaxis CUBE".

Primero, agrega los atributos del select como un GROUP BY estándar estos atributos serán las dimensiones del cubo, luego se agregan las funciones de agregación la cual se aplica sobre las medidas de la dimensión, estas columnas serán llamadas columnas de agregación "TODOS". Si existen N atributos en select, habrán 2^n-1 valores súper agregados.

La figura Nº 3.7 muestra una sentencia SQL para la construcción de un cubo 3D que se ve en la tabla derecha de la figura Nº 3.8 a partir de la tabla izquierda de la misma figura.

```
SELECT Modelo, Año, Color, SUM(ventas) AS Ventas
FROM Ventas
WHERE Modelo in {'Ford', 'Chevy'}
        AND Año BETWEEN 1990 AND 1992
GROUP BY CUBE(Modelo, Año, Color);
```

Figura Nº 3.7 "SQL construcción Cubo Ventas".



Ventas

Modelo	Año	Color	Ventas
Chevy	1990	Rojo	5
Chevy	1990	Blanco	87
Chevy	1990	Azul	62
Chevy	1991	Rojo	54
Chevy	1991	Blanco	95
Chevy	1991	Azul	49
Chevy	1992	Rojo	31
Chevy	1992	Blanco	54
Chevy	1992	Azul	71
Ford	1990	Rojo	64
Ford	1990	Blanco	62
Ford	1990	Azul	63
Ford	1991	Rojo	52
Ford	1991	Blanco	9
Ford	1991	Azul	55
Ford	1992	Rojo	27
Ford	1992	Blanco	62
Ford	1992	Azul	39

Cubo de Datos

Modelo	Año	Color	Ventas
Chevy	1990	Azul	62
Chevy	1990	Rojo	5
Chevy	1990	Blanco	95
Chevy	1990	Todos	154
Chevy	1991	Azul	49
Chevy	1991	Rojo	54
Chevy	1991	Blanco	95
Chevy	1991	Todos	198
Chevy	1992	Azul	71
Chevy	1992	Rojo	31
Chevy	1992	Blanco	54
Chevy	1992	Todos	156
Chevy	Todos	Azul	182
Chevy	Todos	Rojo	90
Chevy	Todos	Blanco	236
Chevy	Todos	Todos	508
Ford	1990	Azul	63
Ford	1990	Rojo	64
Ford	1990	Blanco	62
Ford	1990	Todos	189
Ford	1991	Azul	55
Ford	1991	Rojo	52
Ford	1991	Blanco	9
Ford	1991	Todos	116
Ford	1992	Azul	39
Ford	1992	Rojo	27
Ford	1992	Blanco	62
Ford	1992	Todos	128
Ford	Todos	Azul	157
Ford	Todos	Rojo	143
Ford	Todos	Blanco	133
Ford	Todos	Todos	433
Todos	1990	Azul	125
Todos	1990	Rojo	69
Todos	1990	Blanco	149
Todos	1990	Todos	343
Todos	1991	Azul	106
Todos	1991	Rojo	104
Todos	1991	Blanco	110
Todos	1991	Todos	314
Todos	1992	Azul	110
Todos	1992	Rojo	58
Todos	1992	Blanco	116
Todos	1992	Todos	284
Todos	Todos	Azul	339
Todos	Todos	Rojo	233
Todos	Todos	Blanco	369
Todos	Todos	Todos	941

Figura Nº 3.8 "Resultado Cubo ventas".

Para la tabla de ventas en la Figura Nº 3.8 , tiene 2x3x3 = 18 filas, mientras que el cubo derivado de los datos tiene 3x4x4 = 48 filas. Y los conjuntos respectivos son:

- Modelo.TODOS = TODOS(Modelos) = {Chevy, Ford }
- Año.TODOS = TODOS(Años) = {1990, 1991, 1992}
- Color.TODOS = TODOS(Colores) = {rojo, blanco, azul}

El valor “TODOS” parece ser esencial, pero crea una complejidad sustancial. Es un valor en blanco, como NULL.

La decoración interactúa con los valores agregados. Si la tupla agregada define funcionalmente el valor de la columna de decoración, entonces el valor “TODOS” aparece en la tupla resultante. De otra forma, el campo de decoración es nulo. Por ejemplo la sentencia de la figura Nº 3.9 refleja lo anterior y produce las tuplas de la tabla Nº 3.6.

```
SELECT dia, nacion, MAX(Temp), continente  (nacion)
FROM tiempo
GROUP BY CUBE ( Dia(Time) AS dia,
        Pais (Latitud, Longitud) AS nacion);
```

Figura Nº 3.9 “SQL Decoración”.

Demostrando la Decoración.			
Día	**Nación**	**Max(Temp)**	**Continente**
25/1/1995	USA	28	Norte America
Todos	USA	37	Norte America
25/1/1995	Todos	41	NULL
Todos	Todos	48	NULL

Tabla Nº 3.6 “Resultado de la Decoración”.

La preocupación principal es que a menos que la nación esté presente, el continente no está especificado funcionalmente, por lo tanto es NULL. La función continente de la sentencia SQL de la figura Nº 3.9 devuelve el continente de la nación en el caso que esté especificado, de lo contrario devuelve NULL, en el apartado 3.2.2 se verá la función grouping la cual entrega si el valor NULL es verdadero o falso, esto ayuda a la aparición del valor Todos en la tabla Nº 3.6.

3.2.1.2. Sintaxis del Operador ROLLUP.

Si la aplicación busca sólo un reporte rollup o drill-down, el cubo completo es cortado. Es razonable ofrecer el operador adicional ROLLUP además de CUBE. Ya que ROLLUP produce solamente los siguientes super agregados:

(f1, f2, ..., Todos),
.
.
(f1, TODOS, ..., Todos),
(Todos, Todos, ..., Todos).

Los operadores acumulativos, como la suma o promedio, trabajan especialmente bien con el comando ROLLUP, puesto que el conjunto de respuesta es naturalmente secuencial (lineal), mientras que el CUBE es naturalmente no secuencial (multidimensional). Ambos operadores (ROLLUP y CUBE) deben ser llamados por los operadores acumulativos para aplicarse. En la figura Nº 3.10 se presenta una sintaxis ROLLUP de ejemplo:

```
SELECT Fabrica, Año, Mes, Dia, Color, Modelo,
SUM(precio) AS Ingreso
FROM Ventas
GROUP BY Fabrica,
ROLLUP Año(Time) AS Año,
                        Mes(Time) AS Mes,
                        Dia(Time) AS Dia,
CUBE Color, Modelo;
```

Figura Nº 3.10 "Ejemplo Sintaxis ROLLUP".

3.2.2. Función Grouping.

Es raro ver el valor NULL en el resultado del cubo, este valor en algunos casos representa el valor TODOS, que significa la agrupación de todos los valores de una dimensión. Si el propósito es ayudar al generador de reportes y al software de generación de interfaz gráfica para el usuario (GUI), lo mas simple es implementar la función GROUPING() para discriminar entre los valores NULL y TODOS.

Con la función Grouping() el usuario puede simular el valor TODOS, como se muestra en la figura Nº 3.11:

```
SELECT Modelo, Año, Color, SUM(ventas),
                                GROUPING(Modelo),
                                GROUPING(Año),
                                GROUPING(Color)
FROM Ventas
GROUP BY CUBE(Modelo, Año, Color);
```

Figura Nº 3.11 "Ejemplo Función Grouping".

La suma global será la tupla: (NULL, NULL, NULL, 941, TRUE, TRUE, TRUE). Esta expresión muestra que cada valor NULL representa al valor "TODOS". En el caso que fuera FALSE representaría un valor nulo.

3.2.3. Análisis de los Resultados del Cubo.

CUBE generaliza las agregaciones y el GROUP BY, por lo que todo el análisis de estos resultados, también se puede aplicar como base para analizar el cubo. Sin embargo, analizar las tuplas "TODOS" e implementar los valores "TODOS" no es trivial. Las funciones de agregación disponibles son:

- Distributivas – COUNT(), MIN(), MAX(), SUM().
- Algebraicas – Promedio AVG(), Desviación Estándar STDEV().
- Holísticas – Mediana, Ranking.

Analizar un cubo con funciones de agregación distributivas es relativamente fácil, con funciones algebraicas, una solución eficiente sigue siendo posible.

El cubo de datos se basa en la idea de usar los valores "TODOS" para el GROUP BY y las agregaciones. El operador CUBE es un operador relacional para simplificar la agregación, generalizar agregaciones, y generar tablas cruzadas.

3.3. CONCLUSIÓN DEL TEMA

En los apartados del presente capítulo se analizó el operador Cube y Rollup, este último genera la operación Drill-Up, ambos operadores simulan un cubo multidimensional a partir del cual se pueden implementar las otras operaciones de usuario Dice, Slice, Drill-down y rotación que se detallan en el apartado 2.5.5. y que se implementan en el apartado 4.2.4.1.

Capítulo 4

"Análisis e Implementación del Almacén de Datos"

4. ANÁLISIS E IMPLEMENTACIÓN DEL ALMACÉN DE DATOS.

En el presente capítulo se da a conocer la definición del problema, el análisis de los requisitos y la metodología propuesta para el análisis e implementación del Almacén de Datos detallando cada una de las etapas que componen este proceso. Finalmente, una comparación realizada entre los operadores CUBE Y ROLLUP descritos en el capítulo anterior con el operador GROUP BY.

4.1. DEFINICIÓN DEL PROBLEMA Y ANÁLISIS DE REQUISITOS PARA LA CONSTRUCCIÓN DEL DATAWAREHOUSE.

En una Universidad del país, se cuenta con un sistema de control de alumnos, éste lleva un registro de la vida académica de cada alumno de la universidad. Para nuestro caso y por un tema de accesibilidad a los datos se trabajará sólo con una carrera. El sistema antes mencionado registra entre otras cosas:

- Puntaje de ingreso y año a la carrera.
- Tipo y nombre de colegio de origen del alumno.
- Becas que ha obtenido del alumno durante su carrera.
- Porcentaje de crédito que ha obtenido el alumno durante su carrera.
- Ramos cursados indicando la nota, el profesor y el semestre/año que lo cursó.
- Causales de eliminación indicando semestre, artículo, detalle y estado.
- Para el caso de alumnos titulados se tiene el semestre/año de titulación y su nota.

Durante un trabajo realizado en conjunto con académicos y estudiantes, se determinaron una serie de indicadores que resultan relevantes para procesos de toma de decisiones y de acreditación universitaria, los indicadores obtenidos de estas reuniones son los siguientes, que responden a los requerimientos de los organismos anteriormente mencionados:

- Tiempo promedio de titulación por cohorte: Es el tiempo promedio, medido en años, que demora una generación de estudiantes para titularse. Se entiende por cohorte a una generación de estudiantes que entra en un año específico.

- Tasa de titulación por cohorte: Es el porcentaje de estudiantes que se titula luego de 6 años de estar en la carrera. (Tiempo esperado de titulación para la carrera).

- Tasa de retención por cohorte: Es el porcentaje de estudiantes que pasados los 6 años esperados de titulación aún no terminan la carrera.

- Tasa de aprobación por asignaturas: Porcentaje de aprobación que presenta cada asignatura para un determinado semestre.

- Participación de las mujeres en la matrícula total de la carrera: Porcentaje de estudiantes femeninas por año.

- Cantidad de causales de pérdida por alumno (por género): Número de causales presentadas por varones v/s número de causales presentadas por mujeres, a lo largo de un semestre, año u otro período de tiempo.

- Cantidad de alumnos por región y ciudad de origen: Número de alumnos de la universidad agrupados por región y ciudad de origen.

- Desempeño v/s Becas: Desempeño medido en Asignaturas tomadas v/s Asignaturas Aprobadas, para estudiantes que poseen becas. La idea es determinar si existe algún patrón en aquellos estudiantes que tienen becas, por ejemplo, es esperable que aquellos estudiantes con becas se esfuercen más por tener un buen desempeño, para no perder la beca.

- Desempeño v/s Tipo de colegio de procedencia: Desempeño medido en Asignaturas Aprobadas (y nota de aprobación) en el primer año de estudios en la carrera v/s Colegio de procedencia. Sería interesante determinar como es el desempeño de los estudiantes que provienen de Colegios Privados frente a aquellos que vienen de Colegios Particulares Subvencionados o Públicos.

A partir de estos indicadores se pueden definir los datos de interés.

4.2. ETAPAS PARA LA CONSTRUCCIÓN DEL DW.

Para realizar el proceso de implementación del almacén de datos que dé solución a los requisitos detallados en el apartado anterior se deben llevar a cabo cada una de las etapas graficadas en la figura Nº 4.1.

- **Análisis Base de Datos Fuente:** Descripción del Modelo de Datos Fuente.
- **Diseño Lógico del DW:** Análisis del Almacén de Datos.
  - ✓ Identificar Dimensiones, Hechos y Medidas.
  - ✓ Representar el modelo bajo el Esquema Estrella.
- **Implementación del DW:** Implementación Física del Almacén de Datos, Rolap.
  - ✓ Implementar el Data Warehouse en el Administrador de Base de Datos.
  - ✓ Poblar el Data Warehouse.
  - ✓ Cargar y Mantener el Data Warehouse.
- **Implementación de Cubos:** Creación de Cubos.
  - ✓ Acceso y Análisis de los datos.

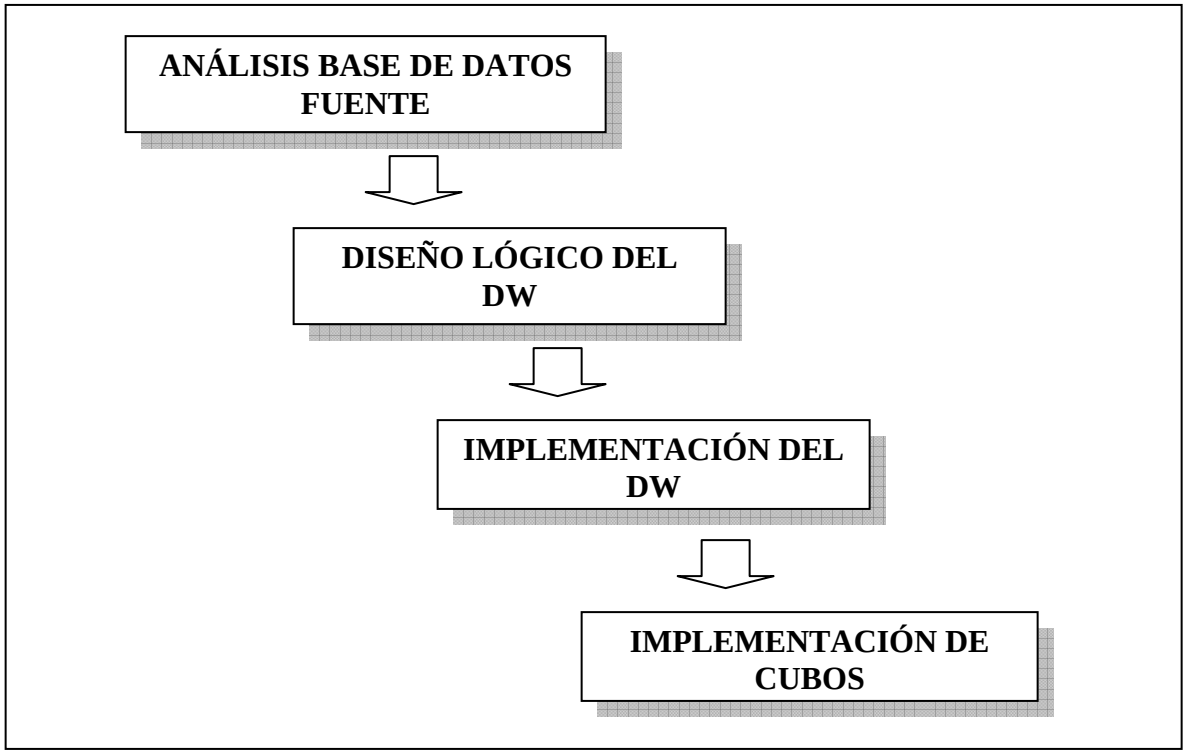


Figura Nº 4.1 "Etapas para la Construcción del DW".

4.2.1. ANÁLISIS BASE DE DATOS FUENTE: Descripción del Modelo de Datos Fuente.

El modelo de datos fuente que se va a utilizar pertenece al sistema académico de una Universidad y corresponde a una parte del modelo de datos completo, basándonos en este último se comienza a construir el modelo del almacén de datos o Data Warehouse.

Una vez analizado el modelo de datos fuente y los requisitos de usuario fue necesario hacer algunas modificaciones ya que habían entidades y atributos que eran innecesarios para la implementación del almacén de datos. Finalmente las tablas que conforman la base de datos fuente y que se crearon en el administrador de base de datos DB2 se muestran en la figura Nº 4.2.

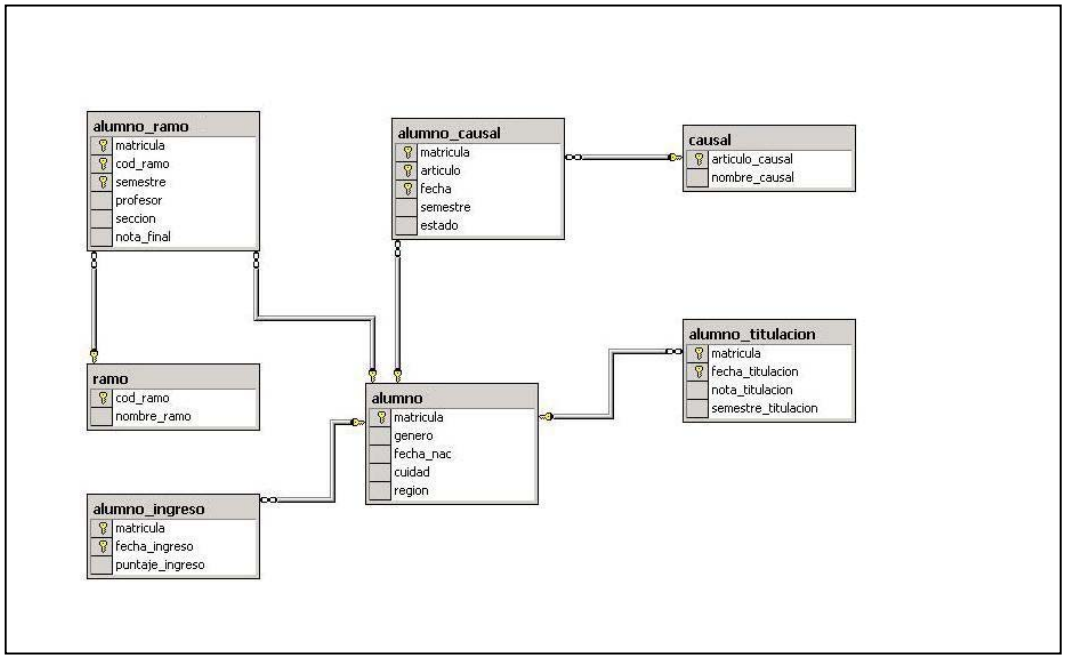


Figura Nº 4.2 "Base de Datos Fuente creada en el Administrador de Base de Datos DB2"

En la tabla N° 4.1 se definen cada una de las entidades de este modelo:

Entidad	Atributo	Descripción Atributo
Alumno		
	matricula	Número de matricula del alumno
	genero	Sexo del alumno
	ciudad	Ciudad de procedencia del alumno
	region	Región de procedencia del alumno
	fecha _nac	Fecha de nacimiento del alumno
Alumno-Ramo	semestre	Identificador de semestre
	matricula	Identificador del alumno
	Cod_ramo	Codigo de ramo
	profesor	Rut profesor del ramo
	Seccion	Sección del ramo
	Nota_final	Nota obtenida en el ramo
Ramo		
	cod_ramo	Identificador de asignatura
	Nombre_ramo	Nombre de la asignatura
Causal		
	articulo_causal	Tipo de causal
	nombre_causal	Nombre causal
Alumno_titulación		
	matricula	Identificador del alumno
	fecha_titulacion	Fecha titulación
	semestre_titulacion	Semestre de titulación
	nota_titulacion	Nota obtenida en la titulación
Alumno_ingreso		
	matricula	Identificador de Alumno
	puntaje_ingreso	Puntaje de ingreso
	fecha_ingreso	Identificador fecha de ingreso
Alumno_causal		
	matricula	Identificador alumno
	articulo	Identificador de causal
	fecha	Fecha de la causal
	semestre	Semestre dela causal
	estado	Aceptada/Rechazada

Tabla N° 4.1 “Descripción Base de Datos Fuente”.

4.2.2. DISEÑO LÓGICO DEL DW: Análisis del Almacén de Datos.

La Base de Datos para el Datawarehouse va a contener los datos provenientes del sistema fuente, será implementada a través de una Base de Datos Relacional que va a representar una base de datos Multidimensional, estructuralmente ambas son iguales ya que se componen por un conjunto de tablas, la diferencia entre éstas está en que las multidimensionales forman agregaciones o resúmenes basándose en las consultas típicas del usuario y están diseñadas para permitir almacenamiento de grandes volúmenes de datos los que pueden ser analizados desde diferentes perspectivas, es por esta razón que su principal aplicación son los Almacenes de Datos [MES2006].

Al representar una Base de Datos multidimensional a través de una Relacional se logra rapidez de acceso, facilidad de presentación, navegación y mantenimiento.

El almacén de Datos va a estar compuesto por tablas de Hechos y Dimensionales que se van a definir a continuación.

4.2.2.1. Identificar Dimensiones, Hechos y Medidas a partir de la Base de Datos Fuente.

Para realizar el proceso de identificación de tablas facts (Hechos) y dimensionales se debe tener especial cuidado con la magnitud de la base de datos, con la información que se tiene y con los requisitos de usuario que se desean responder. Lo principal en esta etapa es tener claramente identificados los datos de interés y los problemas del negocio que deben ser resueltos por el almacén de datos.

Para el caso en cuestión, se creará una tabla Fact para cada uno de los indicadores del apartado Nº 4.2, con sus respectivas dimensiones que describirán una perspectiva bajo la cual el análisis de la Fact puede ser desempeñado. En este caso se modelaron los indicadores con esquema estrella.

Las medidas no se representan como tablas, son atributos de la tabla Fact y serán aquellos indicadores que se desean calcular.

En la tabla Nº 4.2 y 4.3 se describen las Fact y dimensiones como entidades con sus respectivos atributos para el almacén de datos a implementar a partir de la tabla N° 4.2.

Entidad	Atributo	Descripción Atributo
Fact_duracion		
	id_tiempo	Identificador de fecha de ingreso
	matricula	Identificador del alumno
	duracion	Número de años en cursar la carrera
	id_sem_tit	Fecha de titulación
Fact_aprobacion2		
	matricula	Identificador del alumno
	id_asignatura	Identificador de la Asignatura
	nota	Nota obtenida en la signatura
	estado	Identifica si la Asignatura esta Aprobada o Reprobada
Fact_alumnas		
	id_genero	Identificador de genero
	id_tiempo	Identificador de fecha de ingreso
	matricula	Identificador del alumno
Fact_cant_causales	id_genero	Identificador de genero
	id_tiempo	Identificador de fecha de ingreso
	matricula	Identificador del alumno
	cant_causales	Número de Causales de eliminación por Alumno
Fact_alumnos_region	matricula	Identificador del alumno
	id_ciudad	Identificador ciudad de origen del alumno
	cantidad_alumnos	Número de alumnos por región

Tabla Nº 4.2 "Descripción de las tablas Fact para el Almacén de Datos".

Entidad	Atributo	Descripción Atributo
Dim_tiempo		
	Id_tiempo	Identificador de fecha de ingreso
	Id_sem	Fecha de ingreso
	Año	Año de ingreso
Dim_genero		
	Id_genero	Identificador de genero
	Genero	Genero del alumno
Dim_cuidad	Id_ciudad	Identificador de la ciudad de origen
	Nombre_ciudad	Nombre de la ciudad de origen
	Id_region	Identificador de la región de origen
Dim_region	Id_region	Identificador de la región de origen
	Nombre_region	Nombre de la región de origen
Dim_asignatura		
	id_asignatura	Identificador de asignatura
	Nombre	Nombre de la asignatura

Tabla Nº 4.3 "Descripción de las tablas de Dimensión para el Almacén de Datos"**.**

4.2.2.2. Representación del Diseño bajo el Esquema Estrella.

Una vez definidas las tablas Fact y dimensionales, los esquemas estrellas que resultan y que dan soporte a las necesidades de los usuarios se pueden ver en las figuras Nº 4.3 a la Nº 4.7, cada uno de estos atributos se extraen de la base de datos fuente de la figura Nº 4.2.

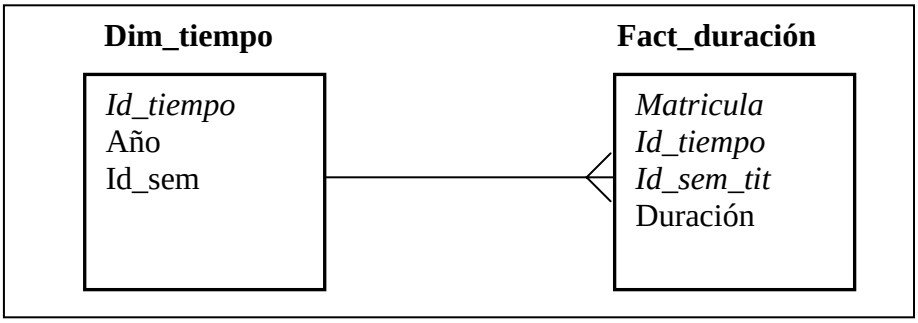


Figura 4.3 "Esquema Estrella Requisitos Tiempo de Titulación, Tasa de titulación y Tasa de retención por cohorte".

Donde (Matricula, Id_tiempo) son la clave compuesta de la tabla Fact_duracón y duración es la medida, (Id_tiempo) es la clave de la dimensión tiempo.

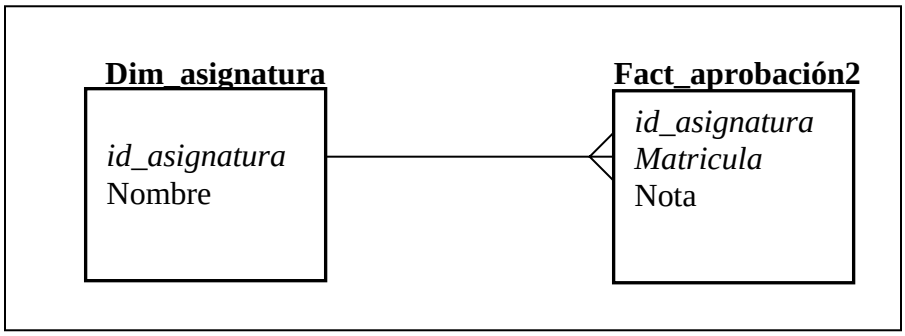


Figura 4.4 "Esquema Estrella Requisito Tasa Aprobación por Asignatura".

Donde (Matricula, Id_tasignatura) son la clave compuesta de la tabla Fact_aprobación2 y nota es la medida, (Id_asignatura) es la clave de la dimensión asignatura.

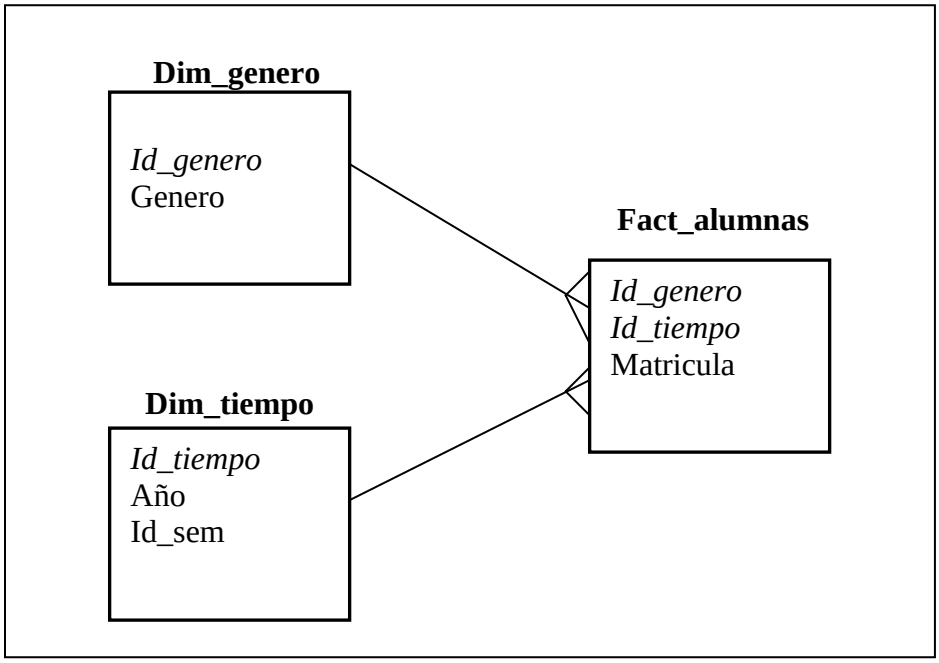


Figura 4.5 "Esquema Estrella Requisito Participación de Mujeres en la Carrera".

Donde (Matricula, Id_genero, Id_tiempo) son la clave compuesta de la tabla Fact_alumnas, (Id_genero) es la clave de la dimensión genero e (Id_tiempo) es la clave de la dimensión tiempo.

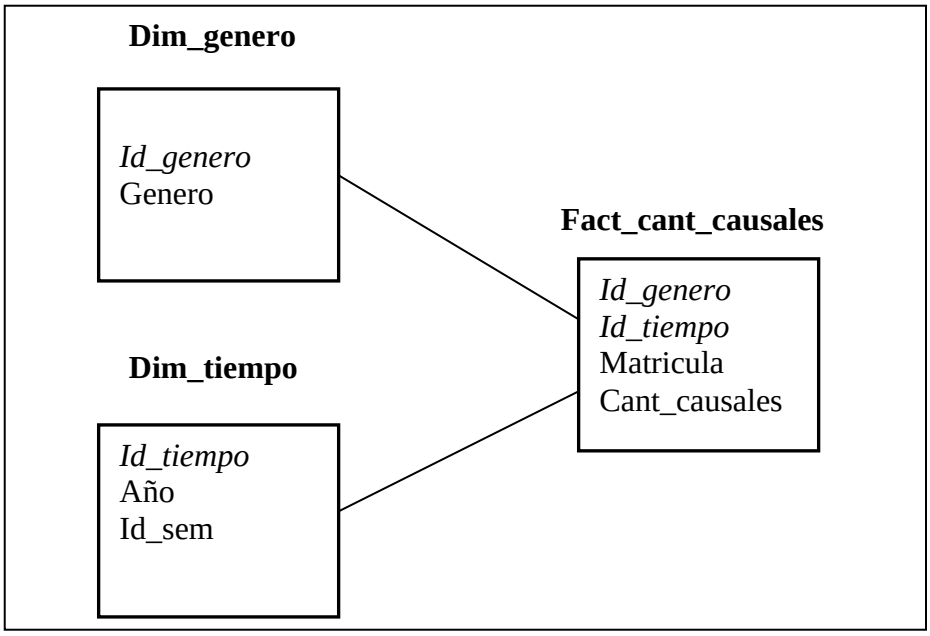


Figura 4.6 "Esquema Estrella Requisito Cantidad de Causales por Alumno y Género".

Donde (Matricula, Id_genero,Id_tiempo) son la clave compuesta de la tabla Fact_cant_cauales, (Id_genero) es la clave de la dimensión genero e (Id_tiempo) es la clave de la dimensión tiempo.

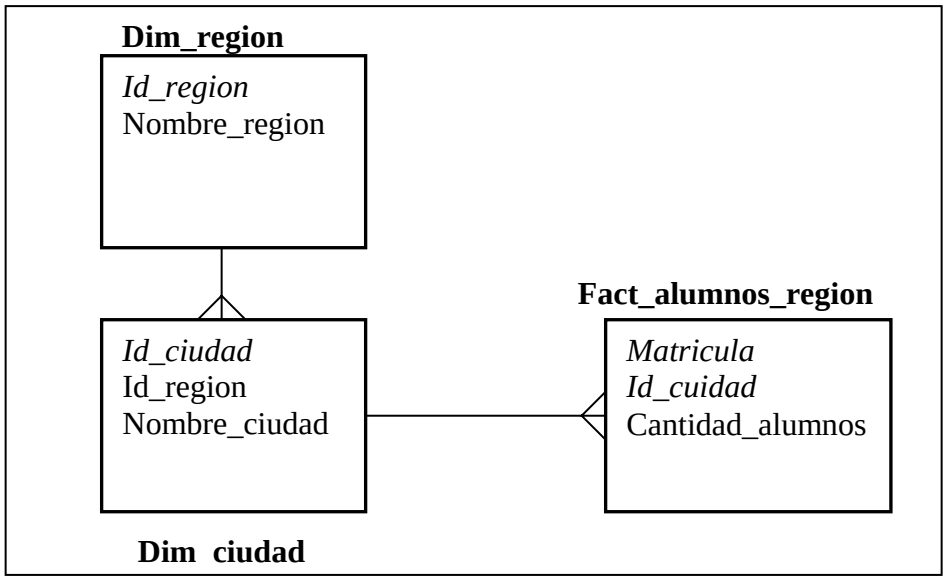


Figura 4.7 "Esquema Estrella Requisito Cantidad de alumnos por región y ciudad de origen".

Donde (matricula, Id_ciudad) son la clave compuesta de la tabla Fact_alumnos_region, (Id_ciudad, Id_region) son la clave de la dimensión ciudad e (id_region) es la clave de la dimensión región.

4.2.3. IMPLEMENTACIÓN DEL DW: Implementación Física del Almacén de Datos, Rolap.

Una vez obtenidos los esquemas multidimensionales (ver apartado Nº 4.3.2), se puede realizar la implementación física del almacén de datos, para lo cual se deben llevar a cabo los siguientes procesos:

4.2.3.1. Creación del Almacén de Datos en el Administrador de Base de Datos

Esta etapa trata de la construcción de las tablas que van a componer el Almacén de Datos, esta tarea se llevó a cabo en DB2 Express versión 9.1 (Para su instalación Ver Anexo B), a través de la ejecución de las siguientes sentencias SQL que se muestran en la tabla Nº 4.4.

Tabla	Sentencia SQL
Fact_duracion	create table fact_duracion(matricula varchar(10) not null, id_tiempo integer, duracion integer, id_sem_tit varchar(10), total_alumnos integer, constraint pk_fact_duracion primary key (matricula), constraint fk_duracion foreign key (id_tiempo) references dim_tiempo (id_tiempo));
Fact_aprobacion2	create table fact_aprobacion(matricula varchar(10) not null, id_asignatura varchar(7) not null, nota decimal(3,2), estado integer, constraint pk_aprobacion primary key (matricula), constraint fk_asignatura foreign key (id_asignatura) references dim_asignatura (id_asignatura));

Tabla Nº 4.4 "Sentencias SQL para Creación de Tablas para el DataWarehouse".

Tabla	Sentencia SQL
Fact_alumnas	create table fact_alumnas(id_genero varchar(1) not null, id_tiempo integer not null, matricula varchar(10) not null, constraint pk_fact_alumnas primary key(matricula), constraint fk_fact_alumnas1 foreign key (id_genero) references dim_genero (id_genero), constraint fk_fact_alumnas2 foreign key (id_tiempo) references dim_tiempo (id_tiempo));
Fact_cant_causales	Create table fact_cant_causales(id_genero varchar(1) not null, id_tiempo integer not null, matricula varchar(10) not null, cant_causales integer, constraint pk_causales primary key(id_genero, id_tiempo, matricula), constraint fk_genero foreign key(id_genero) references dim_genero(id_genero), constraint fk_tiempo foreign key(id_tiempo) references dim_tiempo(id_tiempo));
Fact_alumnos_region	Create table fact_alumnos_region(matricula varchar(10) not null, Id_ciudad varchar(3) not null, constraint pk_region primary key(matricula), constraint fk_ciudad foreign key(id_ciudad) references dim_ciudad(id_ciudad));
Dim_tiempo	create table dim_tiempo(id_tiempo integer not null, id_sem integer, año integer, constraint pk_dim_tiempo primary key (id_tiempo));

Tabla Nº 4.4 (Continuación).

Tabla	Sentencia SQL
Dim_genero	create table dim_genero(id_genero varchar(1) not null, genero varchar(10), constraint pk_dim_genero primary key(id_genero));
Dim_asignatura	create table dim_asignatura(id_asignatura varchar(7) not null, nombre varchar(10), constraint pk_asignatura primary key (id_asignatura));
Dim_ciudad	Create table dim_ciudad(Id_ciudad varchar(3) not null, nombre_ciudad varchar(30) not null, id_region varchar not null(3), constraint pk_ciudad primary key(id_ciudad, id_region), constraint fk_region foreign key(id_region) references dim_region(id_region));
Dim_region	Create table dim_region(Id_region varchar(3) not null, nombre_region varchar(30) not null, constraint pk_region primary key(id_region));

Tabla Nº 4.4 (Continuación).

4.2.3.2. Poblamiento del Almacén de Datos.

El doblamiento de la Base de Datos asociada al Almacén de Datos se refiere al proceso de migración de los datos fuente provenientes de la Base de Datos fuente que se encuentra en la figura Nº 4.2, este proceso comprende extracción, limpieza y transformación de los datos para realizar la carga inicial del Data Warehouse.

El programa encargado de esta tarea se muestra en el Anexo C.

4.2.3.3. Carga y Mantenimiento del Almacén de Datos.

Ya realizado el poblamiento inicial del almacén de datos, se debe desarrollar una aplicación que lleve a cabo la tarea de mantener el Data Warehouse actualizado. Es relevante mencionar que el proceso de extracción se debe realizar cuando los datos de origen no se estén actualizando para evitar inconsistencias.

De este modo la aplicación encargada de este proceso (Ver Anexo C), se ejecutará al inicio y fin de cada semestre.

4.2.4. IMPLEMENTACIÓN DE CUBOS: Creación de Cubos.

La implementación de los Cubos se llevó a cabo a través de sentencias SQL ocupando los operadores CUBE y ROLLUP tratados en el capítulo anterior y que además los proporciona el administrador de base de datos DB2. Estos operadores son una forma simple para simular un cubo. En las figuras Nº 4.9 a la Nº 4.15 se muestran las sentencias SQL que generan los cubos para siete de los nueve indicadores señalados en el apartado Nº 4.1.

- Tiempo promedio de titulación por cohorte.

```
select case grouping(id_sem) when 0 then cast(id_sem as char(4)) when 1 then 'Todos' end as
"Fecha Rendicion" ,avg(duracion)"Promedio Duracion"
from dim_tiempo, fact_duracion
where dim_tiempo.id_tiempo=fact_duracion.id_tiempo and not fact_duracion.duracion is null
group by cube(id_sem)
```

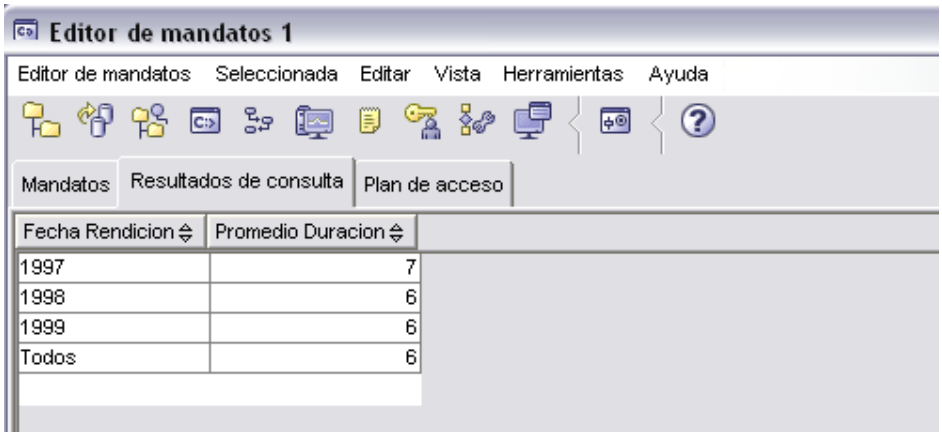


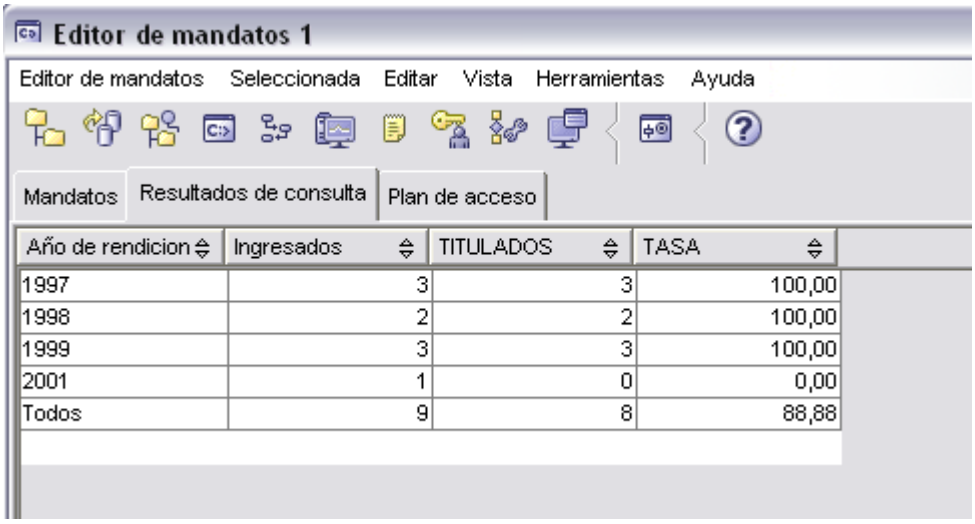
Figura Nº 4.8 "Consulta SQL que simula el Cubo para responder al Indicador Tiempo promedio de titulación por cohorte".

Resultados:

En la figura Nº 4.8, se muestra la sentencia SQL que simula el cubo para responder al indicador "Tiempo promedio de titulación por cohorte", esta sentencia es ejecutada en el editor de mandatos de DB2 (Ver Anexo A). En el resultado se pueden visualizar el promedio de años de titulación para cada año de ingreso, además del promedio para todos los años de ingreso. En esta consulta se ocupó el operador Cube que va a generar un conjunto de resultados que muestra agregados para todas las combinaciones de valores de las columnas seleccionadas, en este caso sólo se seleccionó una columna, por lo tanto, muestra una agregación que es el promedio de duración de todos los años de ingreso. Finalmente, la consulta entrega los datos necesarios y en una forma clara para el proceso de análisis de estos por parte del usuario.

- Tasa de titulación por cohorte.

```
select case grouping(a.id_sem) when 0 then cast(a.id_sem as char(4)) when 1 then 'Todos' end
as "Año de rendicion",
count(*)"Ingresados", count(duracion) Titulados,
cast(cast(count(duracion)*100 as decimal(6,2))/cast(count(*) as decimal(6,2)) as decimal(6,2))
as tasa
from  fact_duracion b left join dim_tiempo a on a.id_tiempo=b.id_tiempo
group by rollup(id_sem)
order by id_sem
```



Año de rendicion	Ingresados	TITULADOS	TASA
1997	3	3	100,00
1998	2	2	100,00
1999	3	3	100,00
2001	1	0	0,00
Todos	9	8	88,88

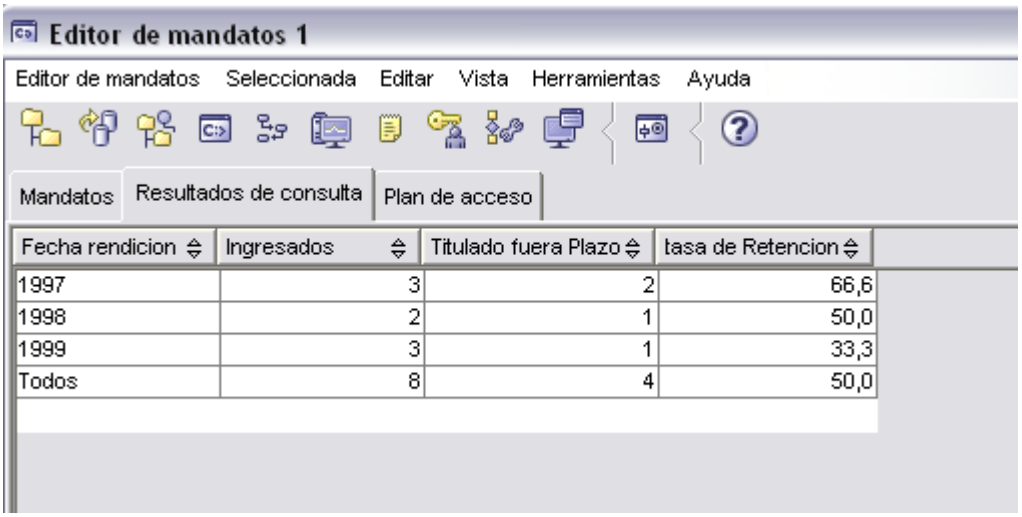
Figura Nº 4.9 "Consulta SQL que simula el Cubo para responder al Indicador Tasa de titulación por cohorte".

Resultados:

En la figura Nº 4.9, se muestra la sentencia SQL que simula el cubo para el indicador "Tasa de titulación por cohorte", ejecutada en el editor de mandatos de DB2. En el resultado se visualiza la cantidad de alumnos ingresados en cada año, cuantos de estos alumnos se han titulado y que porcentaje del total de la matrícula representan. Para la implementación de esta consulta se ocupó el operador Rollup que genera un conjunto de resultados que muestra agregados de las columnas seleccionadas.

- Tasa de retención por cohorte.

```
select case grouping(a.id_sem) when 0 then cast(a.id_sem as char(4)) when 1 then 'Todos' end as
"Fecha rendicion"
,count(id_sem) "Ingresados", count(total_alumnos) "Titulado fuera
Plazo",cast((cast(count(total_alumnos) as decimal(6,2))/cast(count(id_sem) as decimal(6,2)))*100
as decimal(6,1))"tasa de Retencion"
from dim_tiempo a left join fact_duracion b on a.id_tiempo=b.id_tiempo
where length(duracion)<>0
group by rollup(id_sem)
order by id_sem
```



Fecha rendicion	Ingresados	Titulado fuera Plazo	tasa de Retencion
1997	3	2	66,6
1998	2	1	50,0
1999	3	1	33,3
Todos	8	4	50,0

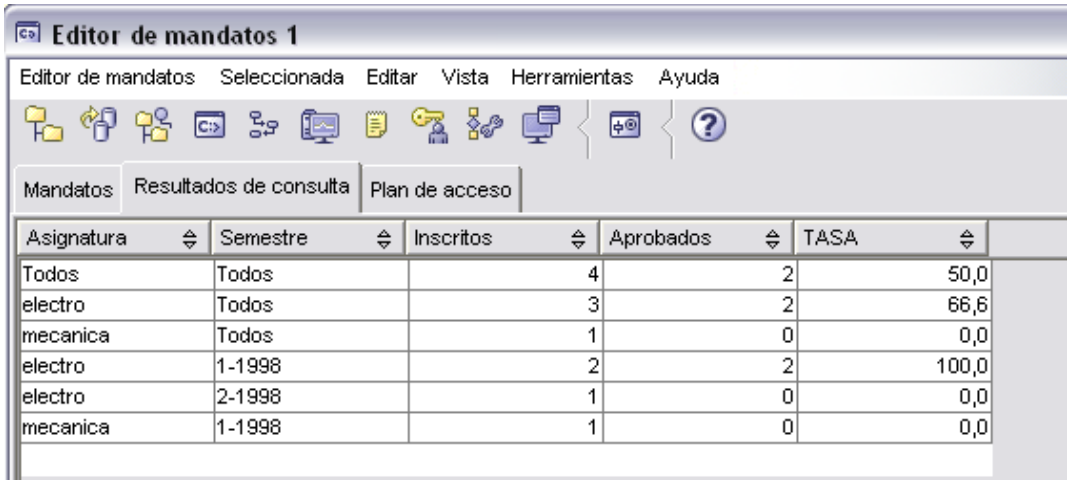
Figura Nº 4.10 "Consulta SQL que simula el Cubo para responder al Indicador Tasa de retención por cohorte".

Resultados:

En la figura Nº 4.10, se muestra la sentencia SQL que simula el cubo para el indicador "Tasa de retención por cohorte", ejecutada en el editor de mandatos de DB2. En el resultado de esta consulta se puede ver el total de ingresados por año de ingreso, la cantidad de estos alumnos que se han titulado, la cantidad promedio de años en que se titularon y la tasa de retención por año de ingreso.

- Tasa de aprobación por Asignatura

```
select  case grouping(nombre) when 0 then nombre when 1 then 'Todos' end as
"Asignatura",
case grouping(semestre) when 0 then semestre when 1 then 'Todos' end as
"Semestre", count(matricula) "Inscritos",
count(estado) "Aprobados",cast((cast(count(estado) as decimal(6,2))/
count(matricula))*100 as decimal(6,1)) as tasa
from dim_asignatura a left join fact_aprobacion2 b on
a.id_asignatura=b.id_asignatura
group by rollup(nombre, semestre)
```



Asignatura	Semestre	Inscritos	Aprobados	TASA
Todos	Todos	4	2	50,0
electro	Todos	3	2	66,6
mecanica	Todos	1	0	0,0
electro	1-1998	2	2	100,0
electro	2-1998	1	0	0,0
mecanica	1-1998	1	0	0,0

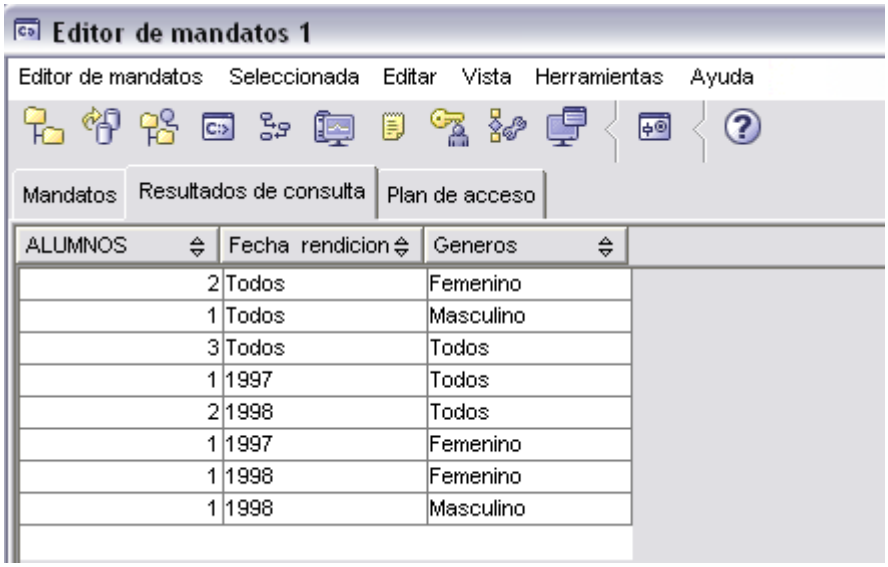
Figura Nº 4.11 "Consulta SQL que simula el Cubo para responder al Indicador Tasa de aprobación por asignatura".

Resultados:

En la figura Nº 4.11, se muestra la sentencia SQL que simula el cubo para el indicador "Tasa de aprobación por asignatura", ejecutada en el editor de mandatos de DB2. En el resultado se puede ver la cantidad de alumnos por asignatura y la tasa de aprobación en cada una de éstas.

- Participación de las mujeres en la matrícula total de la carrera.

```
select count(matricula) as alumnos , case grouping(id_sem) when 0 then cast(id_sem as char(4))
when 1 then 'Todos' end as "Fecha  rendicion", case grouping(genero) when 0 then genero when
1 then 'Todos' end as "Generos"
from fact_alumnas a, dim_genero b, dim_tiempo c
where a.id_tiempo=c.id_tiempo and a.id_genero = b.id_genero
group by cube(id_sem,genero)
```



ALUMNOS	Fecha rendicion	Generos
2	Todos	Femenino
1	Todos	Masculino
3	Todos	Todos
1	1997	Todos
2	1998	Todos
1	1997	Femenino
1	1998	Femenino
1	1998	Masculino

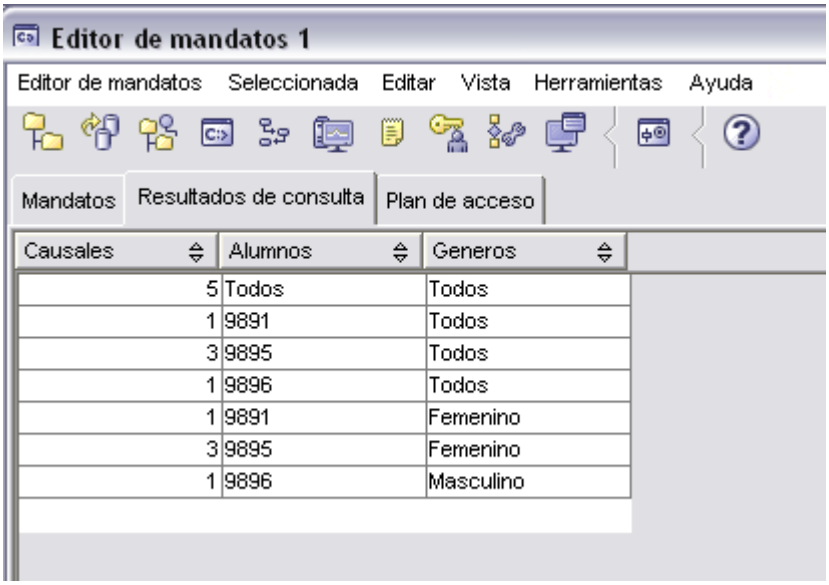
Figura Nº 4.12 "Consulta SQL que simula el Cubo para responder al Indicador Participación de las mujeres en la matrícula total de la carrera".

Resultados:

En la figura Nº 4.12, se muestra la sentencia SQL que simula el cubo para el indicador "Participación de las mujeres en la matrícula total de la carrera", ejecutada en el editor de mandatos de DB2. En esta consulta ocupamos el operador Cube, para este caso se seleccionaron dos columnas, por lo tanto, se generan agregados para todas las combinaciones de estas dos columnas, por lo que el resultado muestra con un mayor nivel de detalle los datos que se quieren obtener. En el resultado se observa la cantidad de alumnos según su género y año de ingreso de una forma clara y fácil para un posterior análisis por parte de los usuarios.

- Cantidad de causales por alumno y género.

```
select sum(cant_causales) as "Causales",case grouping(matricula) when 0 then matricula
when 1 then 'Todos' end as "Alumnos", case grouping(genero) when 0 then genero when
1 then 'Todos' end as "Generos"
from fact_cant_causales a, dim_genero b, dim_tiempo c
where a.id_tiempo=c.id_tiempo and  a.id_genero = b.id_genero
group by rollup(matricula,genero)
```



Causales	Alumnos	Generos
5	Todos	Todos
1	9891	Todos
3	9895	Todos
1	9896	Todos
1	9891	Femenino
3	9895	Femenino
1	9896	Masculino

Figura Nº 4.13 "Consulta SQL que simula el Cubo para responder al Indicador Cantidad de causales por alumno y género".

Resultados:

En la figura Nº 4.13, se muestra la sentencia SQL que simula el cubo para el indicador "Cantidad de causales por alumno y género", ejecutada en el editor de mandatos de DB2. El resultado muestra la cantidad de causales agrupadas por alumno y por género.

- Cantidad de alumnos por región y ciudad de origen.

```
select count(matricula) as "Cantidad de
Alumnos",case grouping (nombre_region) when 0 then nombre_region when 1 then
'Todos' end as "Región" , case grouping (nombre_ciudad) when 0 then  nombre_ciudad
when 1 then 'Todos' end as "Ciudad"
from fact_alumnos_region a,dim_ciudad b,dim_region c
where a.id_ciudad=b.id_ciudad and b.id_region=c.id_region
group by cube (nombre_region,nombre_ciudad )
```

Cantidad deAlumnos	Región	Ciudad
1	Todos	Linares
1	Todos	Los Angeles
1	Todos	Rancagua
1	Todos	San Fernando
1	Todos	Talca
5	Todos	Todos
1	Octava	Todos
2	Septima	Todos
2	Sexta	Todos
1	Octava	Los Angeles
1	Septima	Linares
1	Septima	Talca
1	Sexta	Rancagua
1	Sexta	San Fernando

Figura Nº 4.14 "Consulta SQL que simula el Cubo para responder al Indicador Cantidad de alumnos por región y ciudad de origen".

Resultados:

En la figura Nº 4.14, se muestra la sentencia SQL que simula el cubo para el indicador "Cantidad de alumnos por región y ciudad de origen", ejecutada en el editor de mandatos de DB2. El resultado muestra la cantidad de alumnos agrupados por región y por ciudad de origen.

Todos los resultados anteriores serán base para la creación de Vistas que almacenarán los cubos, sobre estas Vistas se podrá realizar el proceso de análisis por parte de los usuarios. A modo de ejemplo se implementará la Vista que almacenará el cubo para el indicador "Tiempo de titulación por Cohorte" que muestra la figura Nº 4.15 y la Vista que almacenará el cubo para el indicador "Cantidad de alumnos por región y ciudad de origen" que se muestra en la figura Nº 4.16. De la misma forma se implementan las vistas para los demás indicadores.

```
create view CUBO2 as (select case grouping(id_sem) when 0 then cast(id_sem as
char(4)) when 1 then 'Todos' end as "Fecha Rendicion" ,avg(duracion)"Promedio
Duracion"
from dim_tiempo, fact_duracion
where dim_tiempo.id_tiempo=fact_duracion.id_tiempo and not
fact_duracion.duracion is null
group by cube(id_sem))
```

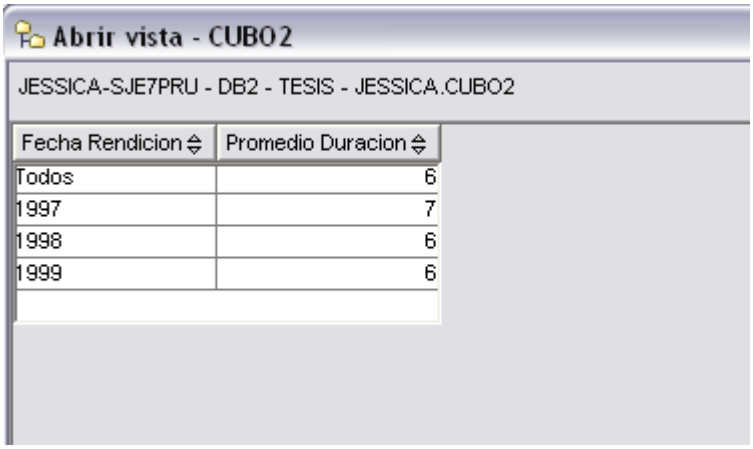
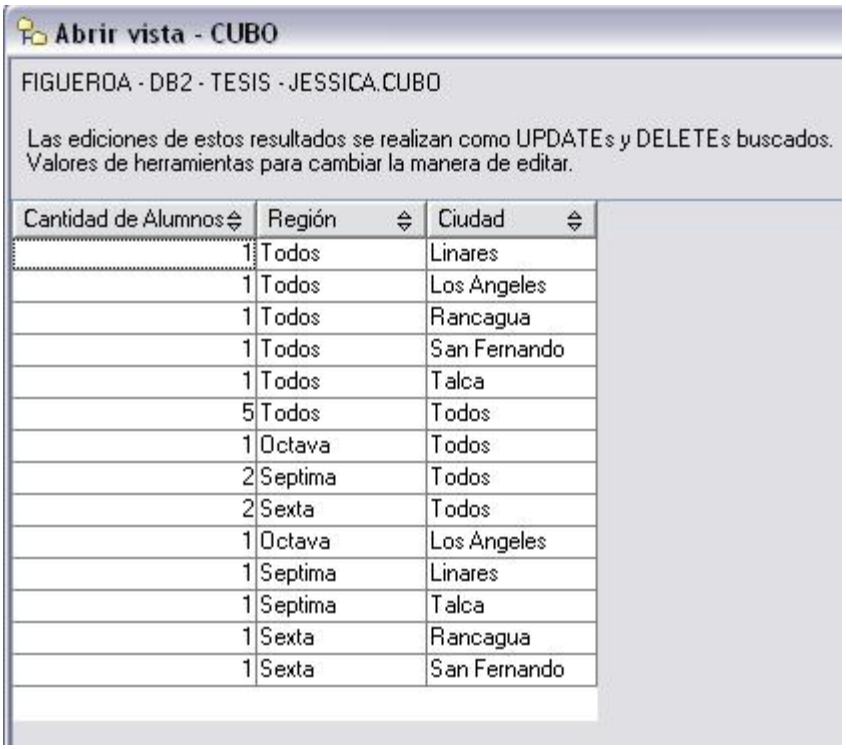


Figura Nº 4.15 "Vista que almacena el Cubo para el indicador Tiempo de titulación por cohorte".

```
create view Cubo as (select count(matricula) as "Cantidad de
Alumnos",case grouping (nombre_region) when 0 then nombre_region when 1
then 'Todos' end as "Región" , case grouping (nombre_ciudad) when 0 then
nombre_ciudad when 1 then 'Todos' end as "Ciudad"
from fact_alumnos_region a,dim_ciudad b,dim_region c
where a.id_ciudad=b.id_ciudad and b.id_region=c.id_region
group by cube (nombre_region,nombre_ciudad ))
```



Cantidad de Alumnos	Región	Ciudad
1	Todos	Linares
1	Todos	Los Angeles
1	Todos	Rancagua
1	Todos	San Fernando
1	Todos	Talca
5	Todos	Todos
1	Octava	Todos
2	Septima	Todos
2	Sexta	Todos
1	Octava	Los Angeles
1	Septima	Linares
1	Septima	Talca
1	Sexta	Rancagua
1	Sexta	San Fernando

Figura Nº 4.16 "Vista que almacena el cubo para el indicador Cantidad de alumnos por región y ciudad de origen".

4.2.4.1. Acceso y Análisis de los Datos.

Ya realizado el poblamiento y carga del Almacén de Datos además de la creación de Vistas que almacenan los Cubos, éste queda listo para ser accedido y poder ejercer labores de análisis sobre los datos, comúnmente la información de un datawarehouse se puede mostrar y representar de variadas formas, la más utilizada es usando un sistema de proceso de análisis en línea (OLAP), el que contiene variadas capacidades de análisis avanzado como el multidimensional, conjuntamente con sofisticadas operaciones de análisis. En este caso el proceso de análisis de los datos se hará sobre las Vistas que almacenan los cubos a través de consultas SQL estándar que generarán las operaciones que se pueden realizar sobre éstos, dichas operaciones son Dice, Slice, Drill-up, Drill-down y Rotación tratada en el apartado Nº 2.5, a continuación se implementarán cada una de éstas.

Al implementar la operación Dice, la cual fija valores para una dimensión específica, se obtiene como resultado los datos que se muestran en la figura Nº 4.17 en donde con una sentencia SQL simple se puede implementar esta operación y obtener el tiempo promedio de salida de la carrera para los años de ingreso 1998 y 1999 a partir del Cubo de la figura Nº 4.15.

Definición Algebraica:

$$\sigma\left(C^{1}_{cubo2},\ fecha\_rendicion = '1998'\ or fecha\_rendicion = '1999'\right)$$

```
select *
from CUBO2
where "Fecha Rendicion"= '1998' or "Fecha Rendicion" = '1999'
```

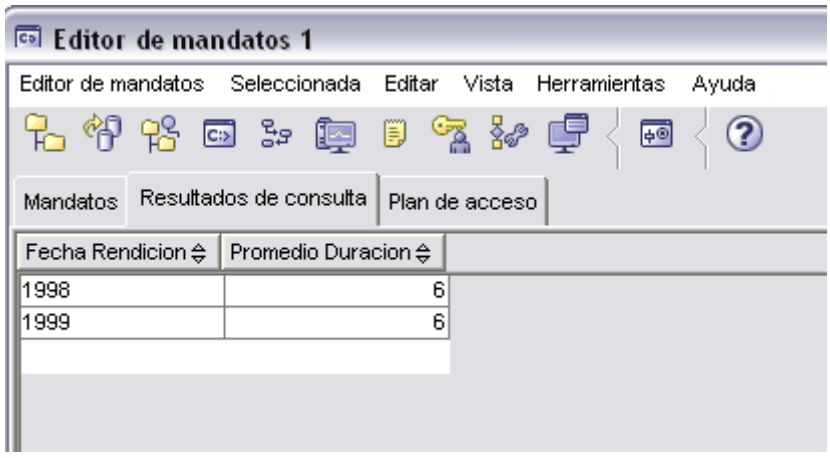


Figura Nº 4.17 "Ejemplo Operación Dice".

Al implementar la operación Slice, la cual restringe los valores que consideramos en las dimensiones según alguna condición, se obtiene como resultado los datos que se muestran en la figura Nº 4.18 en donde con una sentencia SQL simple se puede implementar esta operación y obtener el tiempo promedio de salida de la carrera para todos los años de ingreso a partir del Cubo de la figura Nº 4.15.

Definición Algebraica: $\prod\left(C^{1}_{cubo2}, fecha\_rendicion = 'Todos'\right)$

```
select *
from CUBO2
where "Fecha Rendicion"= 'Todos'
```

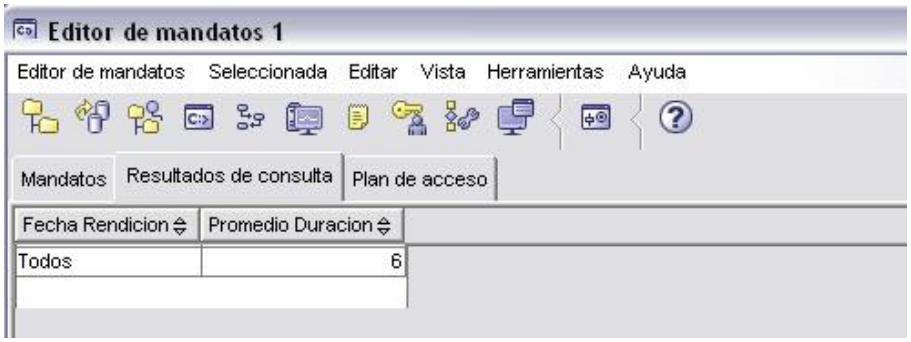
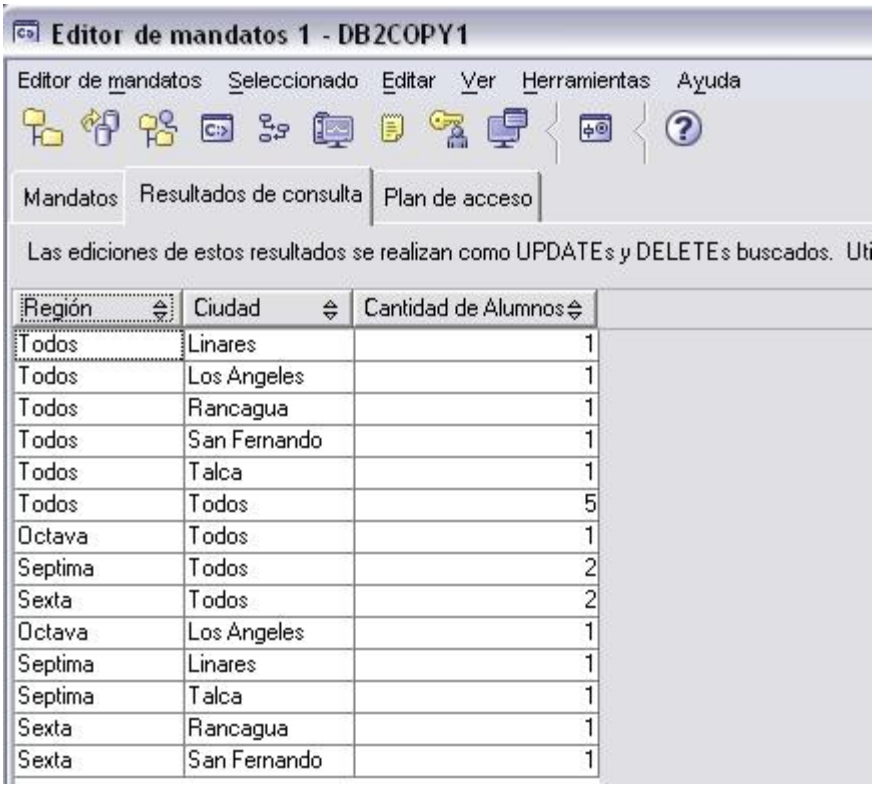


Figura Nº 4.18 "Ejemplo Operación Slice".

Al implementar la operación Drill-down que significa cambiar el nivel de definición de los hechos a niveles inferiores de las jerarquías, se obtiene como resultado los datos que se muestran en la figura Nº 4.19 en donde con una sentencia SQL simple se puede implementar esta operación y obtener el número de alumnos agrupados por región y ciudad de origen a partir del Cubo de la figura Nº 4.16.

Definición Algebraica: $\Downarrow \left(C^{1}_{cubo}, cuidad\right) = C^{'2}_{cubo'}\left(cuidad, region\right)$

```
select "Región","Ciudad", "Cantidad de Alumnos"
from cubo
```



Región	Ciudad	Cantidad de Alumnos
Todos	Linares	1
Todos	Los Angeles	1
Todos	Rancagua	1
Todos	San Fernando	1
Todos	Talca	1
Todos	Todos	5
Octava	Todos	1
Septima	Todos	2
Sexta	Todos	2
Octava	Los Angeles	1
Septima	Linares	1
Septima	Talca	1
Sexta	Rancagua	1
Sexta	San Fernando	1

Figura Nº 4.19 "Ejemplo Operación Drill-down".

Al implementar la operación Drill-up que significa subir en la jerarquía, se obtiene como resultado los datos que se muestran en la figura Nº 4.20 en donde con una sentencia SQL simple se puede implementar esta operación y obtener el número de alumnos agrupados por región de origen a partir del Cubo de la figura Nº 4.16.

Definición Algebraica: $\Uparrow \left(C^{2}_{cubo}, ciudad, region\right) = C^{'1}_{cubo'}\left(ciudad\right)$

```
select *
from cubo
where "Ciudad"='Todos'
```

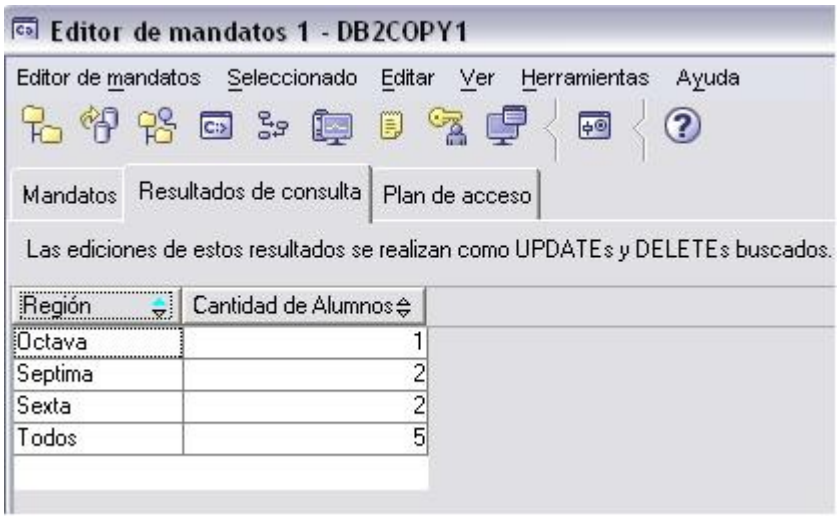


Figura Nº 4.20 "Ejemplo Operación Drill-up".

4.3. FASE DE COMPARACIÓN.

En el presente apartado se hace una comparación de las consultas implementadas en la sección 4.2.4 en las cuales se ocuparon los operadores CUBE y ROLLUP para simular cubos, con consultas en las cuales sólo se ocuparon las cláusulas Group By para realizar análisis sobre los datos del Almacén de Datos. Aquí se pretende demostrar que usando estos operadores descritos en el capitulo 3 se puede obtener una información más detallada y en una forma más entendible para el usuario, además de obtener ventajas en el tiempo de respuesta.

Para efectos de comparación sólo se implementarán dos de las consultas anteriormente realizadas.

- Tiempo promedio de titulación por cohorte (Consulta realizada sin los operadores).

```
select avg(duracion)"Promedio Duracion", id_sem "Año Ingreso",  (
                    select sum(duracion)/count(duracion)
                    from fact_duracion ) as total
from dim_tiempo, fact_duracion
where dim_tiempo.id_tiempo=fact_duracion.id_tiempo
group by id_sem
order by id_sem
```

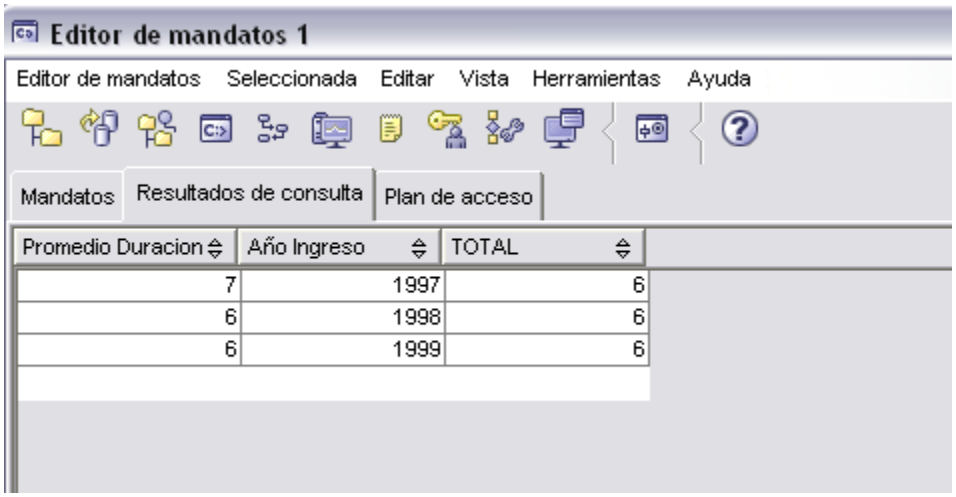


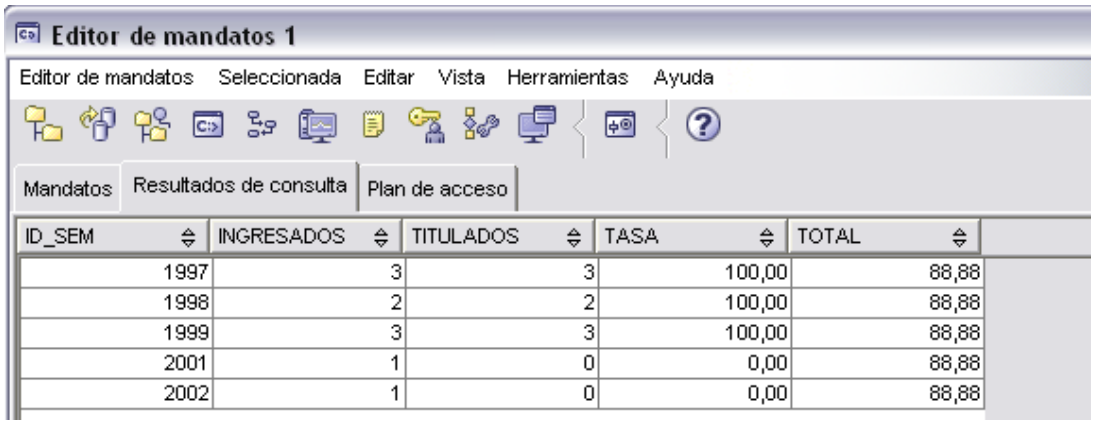
Figura Nº 4.21 "Consulta SQL y Resultado para indicador Tiempo promedio de titulación por cohorte (sin operadores)".

Resultados:

Ambas consultas retornan los mismos resultados. En la sección 4.2.4 en la figura Nº 4.8 se muestran los resultados de la misma consulta realizada usando los operadores. A simple vista se puede ver que ese resultado es mucho más fácil para poder realizar un análisis por el usuario, los datos están mostrados de tal forma que se entienden claramente. A diferencia de los resultados que se ven en la figura Nº 4.21 en donde la consulta esta hecha sin ocupar los operadores, aquí los datos son mostrados en una forma que es difícil de ser entendida por el usuario que desea hacer el análisis. Otra diferencia se ve en la sentencia SQL que se debe ejecutar para obtener los datos, claramente observamos que para la segunda consulta se debe realizar una sentencia SQL mas compleja ocupando select anidados, mientras que en la primera consulta la sentencia que se ejecuta es más sencilla.

- Tasa de titulación por cohorte (Consulta realizada sin los operadores).

```
select a.id_sem ,
count(*) as ingresados, count(duracion) Titulados,
cast(cast(count(duracion)*100 as decimal(6,2))/cast(count(*) as decimal(6,2)) as
decimal(6,2)) as tasa,
cast((select cast(count(matricula) as decimal(6,2)) from fact_duracion where
length(duracion)<>0)*100/(select cast(count(matricula) as decimal(6,2)) from
fact_duracion ) as decimal(6,2)) as TOTAL
from dim_tiempo a left join fact_duracion b on a.id_tiempo=b.id_tiempo
group by id_sem
order by id_sem
```



ID_SEM	INGRESADOS	TITULADOS	TASA	TOTAL
1997	3	3	100,00	88,88
1998	2	2	100,00	88,88
1999	3	3	100,00	88,88
2001	1	0	0,00	88,88
2002	1	0	0,00	88,88

Figura Nº 4.22 "Consulta SQL y Resultado para indicador Tasa de titulación por cohorte (sin operadores)".

Resultados:

Al igual que en la consulta anterior, de los resultados de la consulta realizada ocupando los operadores que se muestra en la figura Nº 4.9 se puede observar que éstos son perfectamente entendibles y facilita el proceso de análisis del usuario, a diferencia de la consulta hecha sin los operadores (figura Nº 4.22), en donde los resultados son confusos y dificultan el análisis de los datos. Además, claramente el número de sentencias SQL que se debe ejecutar para la consulta hecha con los operadores es mucho menor y más sencilla, mientras que en la consulta en donde no se ocupan los operadores el número de sentencias SQL es mayor y la consulta en sí es más compleja.

TABLA COMPARATIVA		
IMPLEMENTACIÓN DE INDICADORES	**Con Operadores Cube y Rollup**	**Sin Operadores**
Tiempo de Ejecución	Menor tiempo de ejecución.	Mayor tiempo de ejecución.
Cantidad de Sentencias	Menor cantidad de sentencias SQL.	Mayor cantidad de sentencia SQL.
Complejidad de la Sentencia	Menor complejidad de la sentencia SQL.	Mayor complejidad de la sentencia SQL.
Facilidad de Entendimiento Resultados.	Mayor Entendimiento de los Datos de Salida.	Menor Entendimiento de los Datos de Salida.

Tabla Nº 4.5 "Tabla comparativa de la implementación de indicadores".

Capítulo 5

“Conclusión”

5. CONCLUSIÓN

Al finalizar este trabajo se puede concluir en primer lugar que para implementar un almacén de datos o DW como una solución para el proceso de análisis de datos se debe tener muy claro cual es el proceso de negocio que se desea apoyar con esta solución y cuales son las necesidades de los usuarios. En este trabajo, como ya se mencionó en el capitulo 4, el DW debe responder a indicadores que sirven para el proceso de acreditación universitaria.

Una vez determinado el proceso de negocio y los requisitos de los usuarios, se implementó un Data Warehouse utilizando el administrador de base de datos relacional DB2, ocupando instrucciones SQL que este administrador proporciona y que simulan cubos multidimensionales indispensables para el proceso de análisis de información. Se escogió este administrador por dos razones: la primera de ellas es porque no se ha trabajado anteriormente con este DBMS (Data Base Management System) en la universidad, por lo que es un desafío importante el poder estudiar su desempeño, y en segundo lugar porque proporciona operadores que simulan cubos multidimensionales, que como ya se mencionó anteriormente, permiten implementar un DW sólo con instrucciones SQL, sin tener que recurrir a la herramienta Case que este administrador posee para la generación de cubos.

Las etapas que se aplicaron en la implementación son las que comúnmente se deben aplicar a cualquier proceso de implementación de un Almacén de Datos, pero con una mínima diferencia, como primera etapa se llevo a cabo el análisis de la base de datos fuente, luego el diseño lógico identificando medidas, dimensiones y hechos, para posteriormente generar las tablas que componen el almacén de datos en el administrador de base de datos realizando una implementación Rolap. Es en esta etapa donde se produce la diferencia, ya que para el proceso normal de implementación de cubos mediante una herramienta que los genere automáticamente, esta fase se hace transparente para el desarrollador. En este caso como los cubos multidimensionales se simularon sólo con instrucciones SQL estándar ocupando para esto el operador Cube, esta etapa no es transparente. Finalmente, termina el proceso de desarrollo realizando la implementación de los cubos que posteriormente se almacenan en vistas.

La solución implementada es una alternativa “free” para la implementación de cubos multidimensionales, ya que se trabajó con el administrador de Base de datos DB2 Express Versión 9.1, que es la versión libre de este administrador y con instrucciones SQL, esta solución se puede implementar perfectamente en empresas pequeñas en donde el adquirir nuevas herramientas o tecnologías es más complejo, y en cuyo caso el nivel de detalle de la información que se necesita no es demasiado específico.

Una vez implementado el almacén de datos, se puede decir que se han combinado dos tecnologías, una relacional para las consultas que simulan los cubos ya que el modelo se basa en una estructura de tablas y relaciones, con una multidimensional para el análisis de la información, además el DW responde absolutamente a los indicadores detallados en el apartado 4.1, ya que entrega claramente la información que los usuarios necesitan, por lo tanto se puede decir que la solución propuesta es efectiva y que sólo ocupando sentencias SQL de DB2 se logró cumplir con el objetivo de implementar un Data Warehouse como una solución Rolap..

El proceso de instalación de DB2 (Ver Anexo B), es completamente intuitivo, puesto que es guiado por un asistente de instalación que facilita el proceso para el usuario.

Finalmente, como trabajo futuro se recomienda la implementación de un almacén de datos utilizando la herramienta Case (Cube View 8.1), que posee DB2 que permita ver las potencialidad de esta herramienta.

BIBLIOGRAFÍA

[BIT2005] BITAM, “Business Intelligence”. [en línea].
<http://www.bitam.com/spanish/AcercaDeBI.htm>
[Consulta: Noviembre 2005].

[CAS2001] CASARES, CLAUDIO,
http://www.programacion.com/bbdd/tutorial/warehouse/3/,
Bases datos en castellano, 2001.

[HER2003] HERNANDEZ, JOSE,
http://www.dsic.upv.es/~jorallo/cursoDWDM/dwdm-I.pdf
Departamento de Sistemas Informáticos y Computación Universidad Politécnica de Valencia, España.
[Consulta: Noviembre 2005]

[IBM2005] IBM, Sitio Oficial. [en línea]
http://www.software.ibm.com/data/db2/udb
[Consulta: Noviembre 2005]

[INM2002] INMON, W.H,
Building the Data Warehouse,
Tercera Edición, John Wiley & Sons, 2002.

[INM1994] INMON, W. H.; HACKATHORN, RICHARD D. 1994.
“Using the Data Warehouse”. New York: John Wiley & Sons.
ISBN: 0-471-05966-8.

[JGR1996] J. GRAY, A. BOSWORTH, A. LAYMAN, H. PIRAHESH,
“Data cube: a relational aggregation operator generalizing group-by,.cross-tabs and subtotals”
Int'l Conf. on Data Engineering '96. Technical report

[JGO1998] GUSAETA GONZÁLEZ JORGE,
http://www.iee.com/tendencias-tecnológicas.html,
Boletín IEE “Tendencias Tecnológicas”, 1998

[KIM1996] KIMBALL, RALPH,
The Data Warehouse Toolkit,
http://www.dwinfocenter.org/ defined.html, 1996.

[LFD2003] LING FENG, THARAM DILLON,
Using fuzzy linguistic representations to provide explanatory semantics for Data Warehouse,

[MES2006] MESA CARLOS, VILLARROEL MARÍA
http://www.dcc.uchile.cl/cmesa/ccb2c/info_pre_3.html
“Modelamiento Multidimensional”
[Consulta: Mayo 2006].

[OST1993] OSTTERFELDT SUSAN,
"Data Warehousing, Data Modeling and Desing",
MicroStrategy Education, nov 96.

[PER2001] PERALTA, VERONICA,
http://www.fing.edu.uy/~vperalta/,
Facultad de Ingeniería, Universidad de la República, Montevideo, URUGUAY.

[SAS2003] SAS INSTITUTE EUROPEAN OFFICE, [en línea]
http://www.sas.com,
SAS Institud Inc. World Headquartes
[Consulta: Diciembre 2005]

ANEXO A

“Ambiente de Trabajo de DB2”

A. AMBIENTE DE TRABAJO PARA DB2.

A.1. Centro de Control

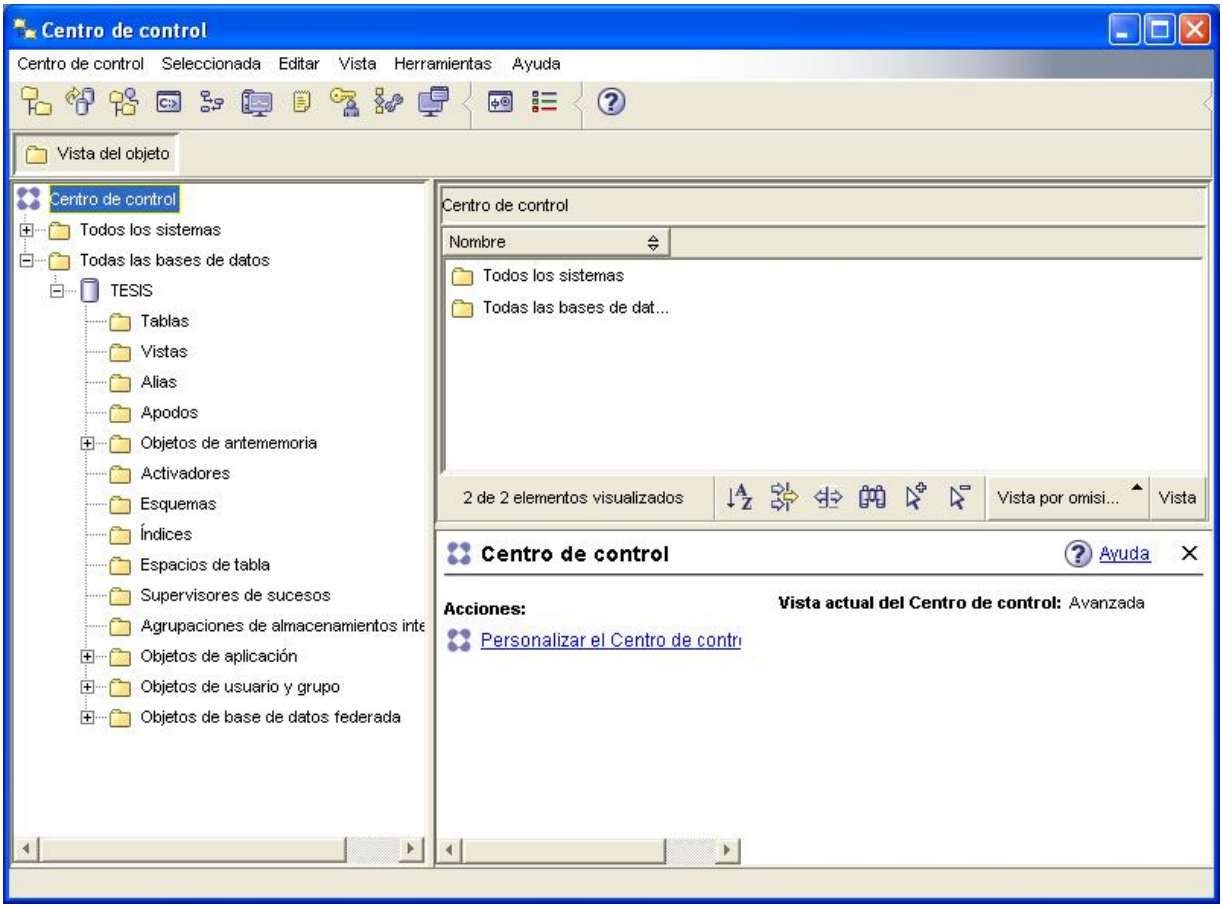


Figura A.1 "Centro de Control de DB2".

En el centro de control se pueden visualizar todos los objetos y acciones disponibles para el usuario.

A.2. Editor de Mandatos

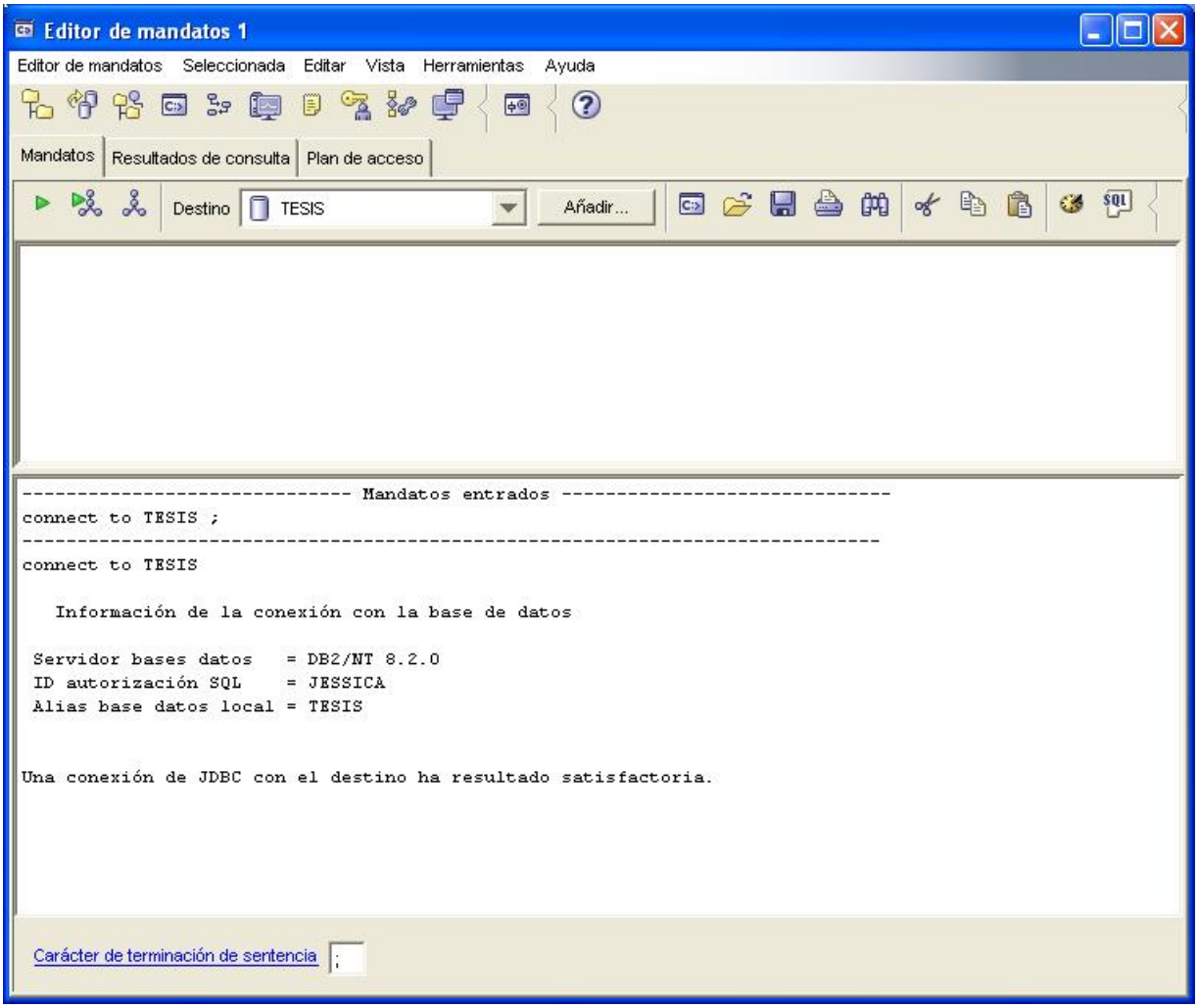


Figura A.2 "Editor de Mandatos de DB2".

En el editor de mandatos de de DB2 se ejecutan las sentencias SQL.

A.3. Ejecución de Consulta

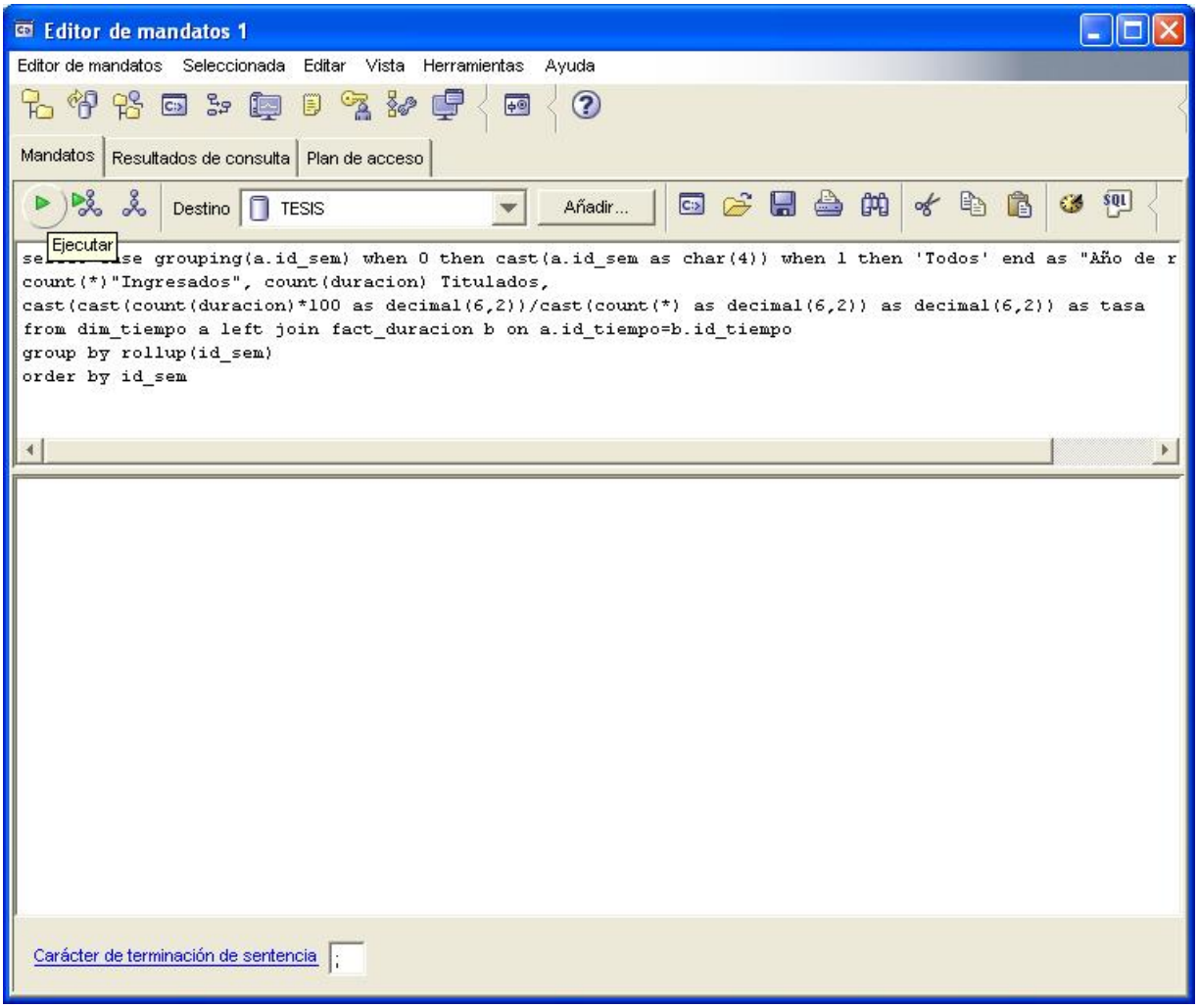


Figura A.3 "Editor listo para ejecutar consulta".

Haciendo click en el botón ejecutar se lleva a cabo la ejecución de la sentencia Sql.

A.4. Resultado Consulta

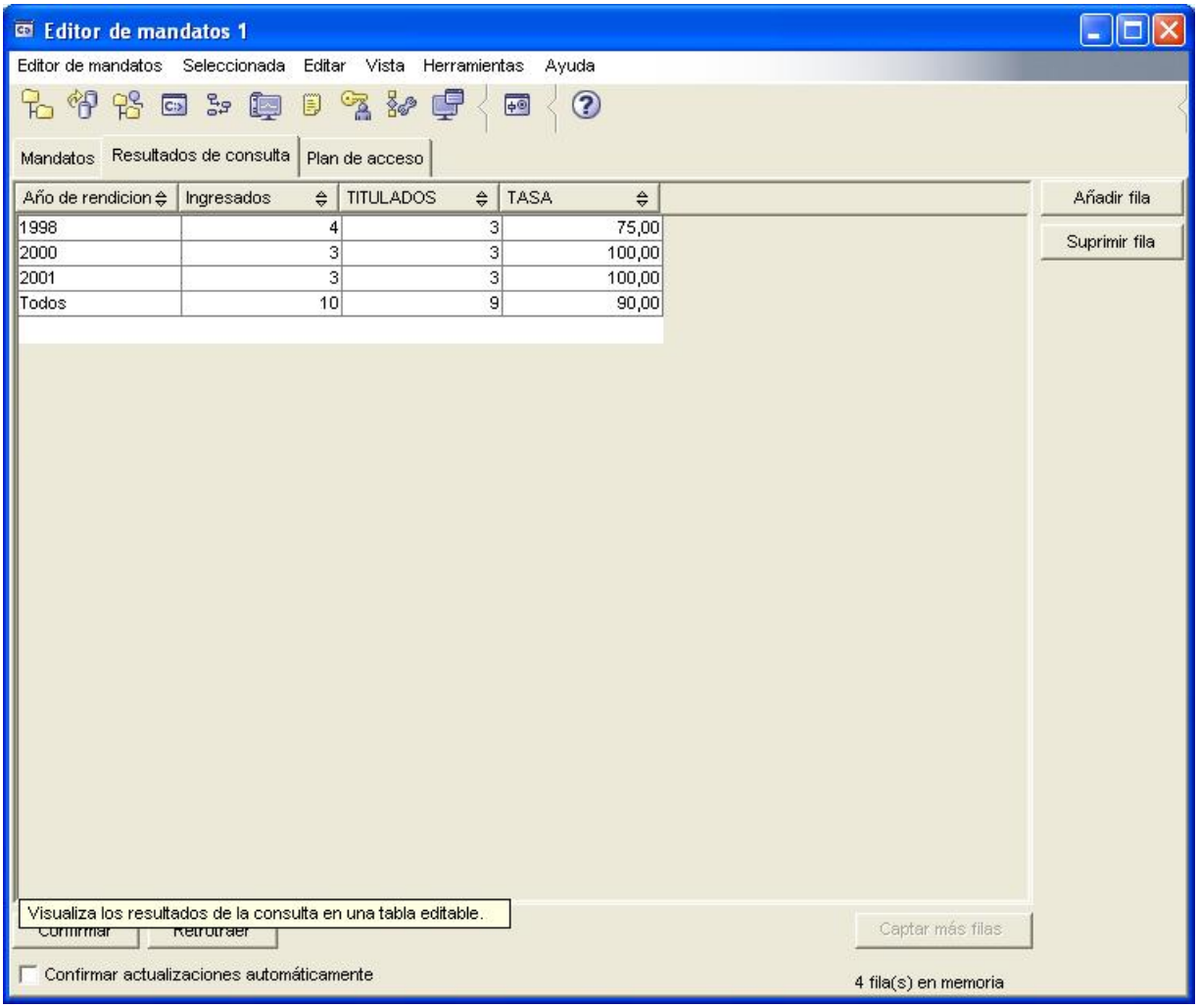


Figura A.4 "Visualizador de Resultados".

Luego de realizada la ejecución de la consulta se visualizan los resultados en una tabla.

A.5. Asistente para Crear Base de Datos

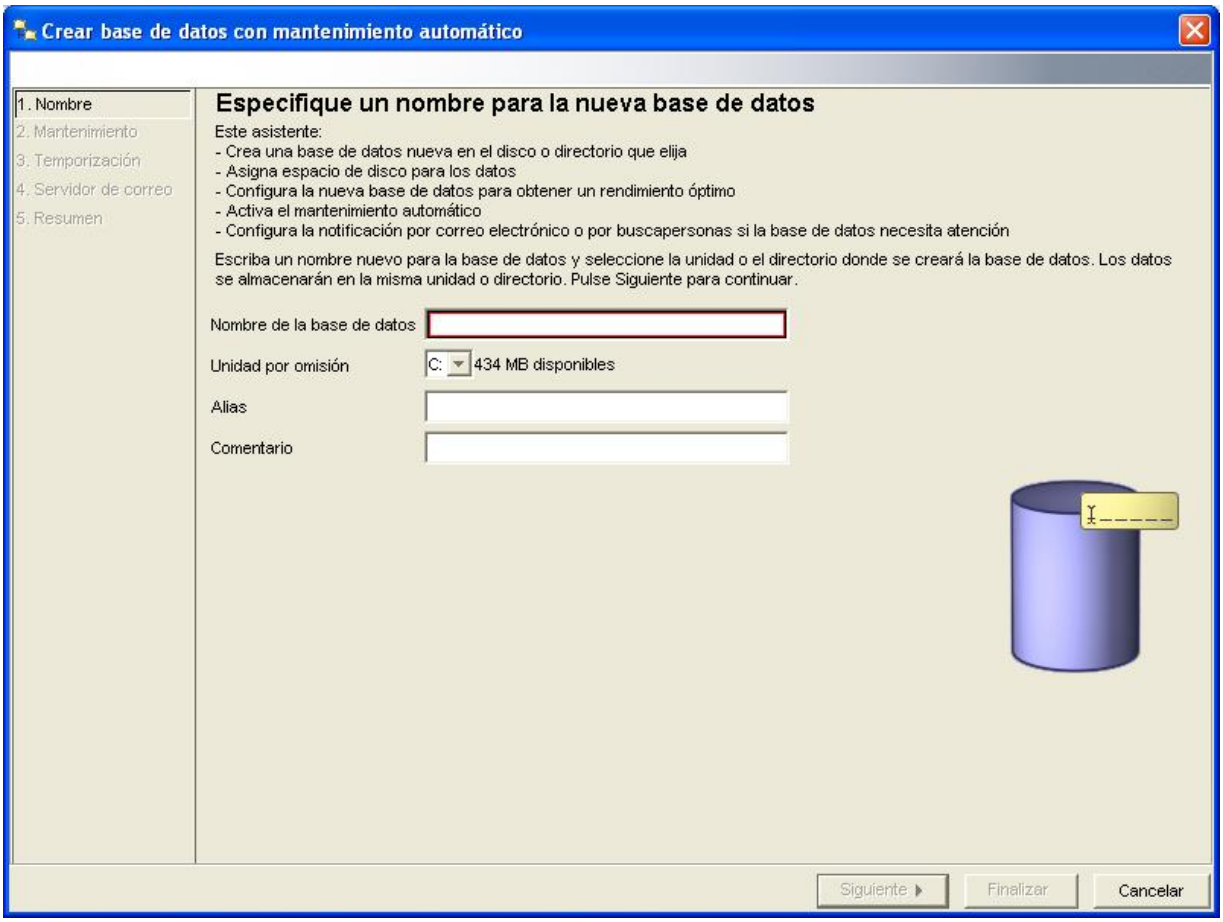


Figura A.5 "Asistente para crear nueva BD".

DB2, cuenta con un asistente que permite guiar al usuario en el proceso de creación de una nueva base de datos.

A.6. Asistente para Crear Tablas

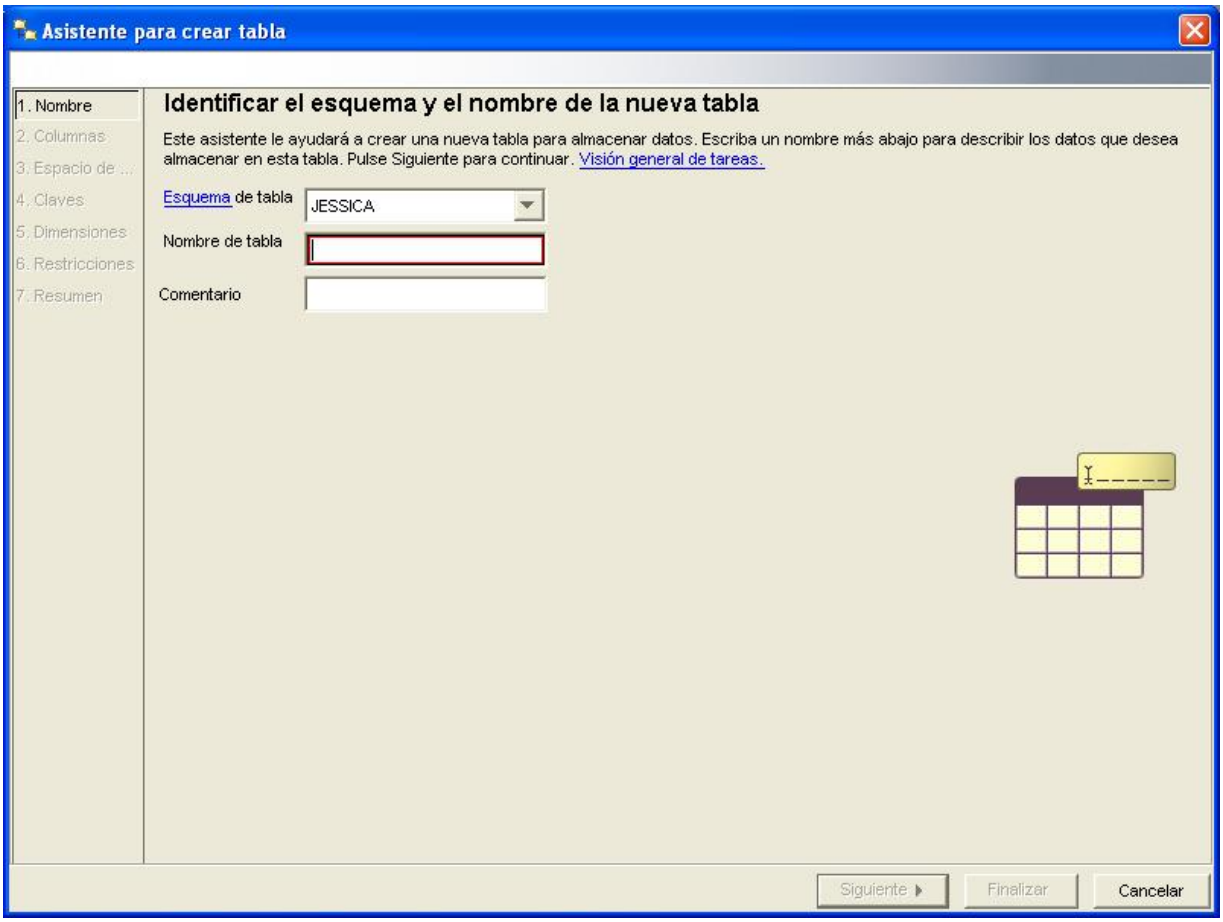


Figura A.6 "Asistente para crear Nuevas tablas".

DB2 cuenta con un asistente que permite guiar al usuario en el proceso de creación de una tabla en el caso que éste no desee hacer este proceso con sentencias Sql en el editor de mandatos (ver figura A.2).

ANEXO B

“Configuración e Instalación de DB2”

B. CONFIGURACIÓN E INSTALACIÓN DE DB2.

B.1. Configuración e Instalación de DB2

A continuación se mostrarán los pasos que se deben seguir para la instalación del Software de Gestión de Información DB2 Express, Versión 9.1.

Como primer paso, se despliega una ventana con información general del software, ofreciendo además las opciones para su instalación, ver figura B.1.

Figura B.1 "Información de DB2".

En la figura B.2, se aprecia el mensaje de bienvenida que ofrece el sistema al iniciar su instalación.

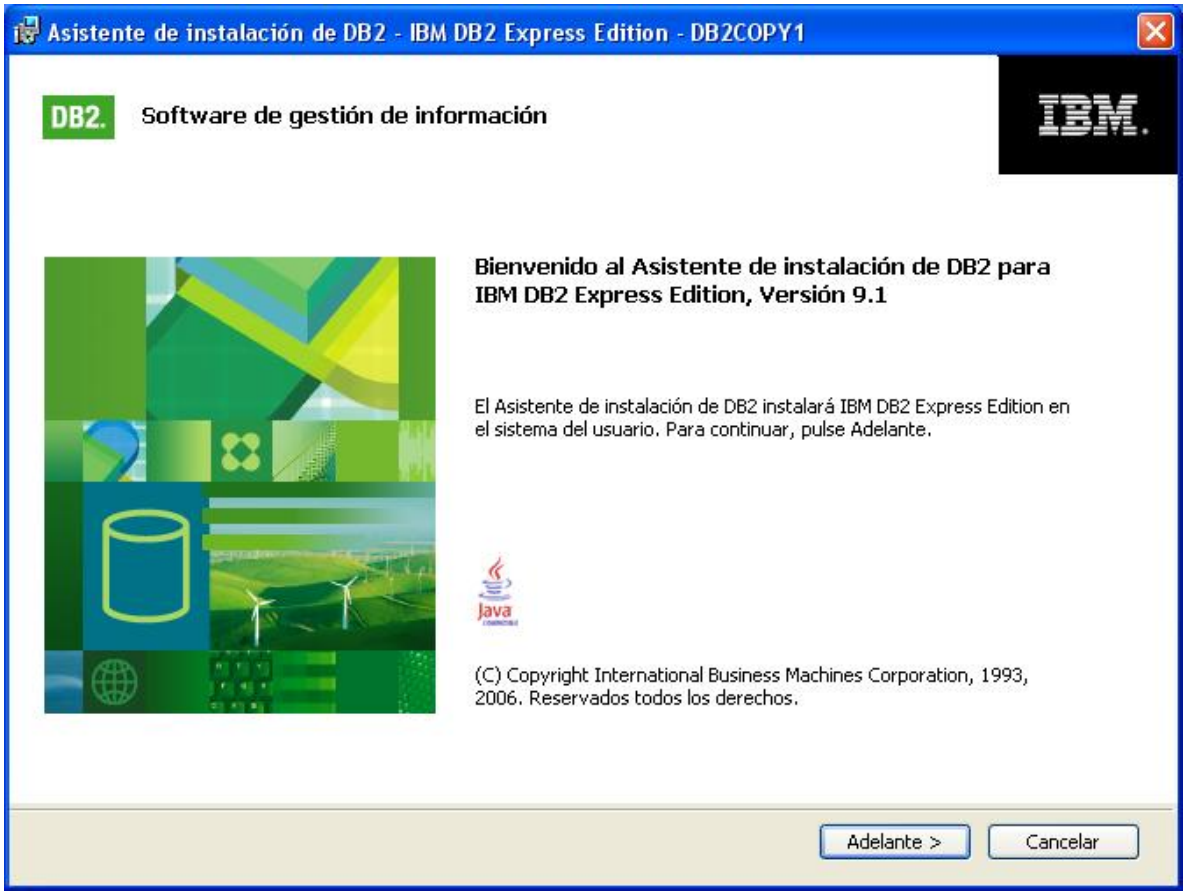


Figura B.2 "Asistente de Instalación".

Como siguiente opción, se despliega el contrato de licencia del software, el cual debe ser aceptado para poder seguir con la instalación, como se muestra en la figura B.3.

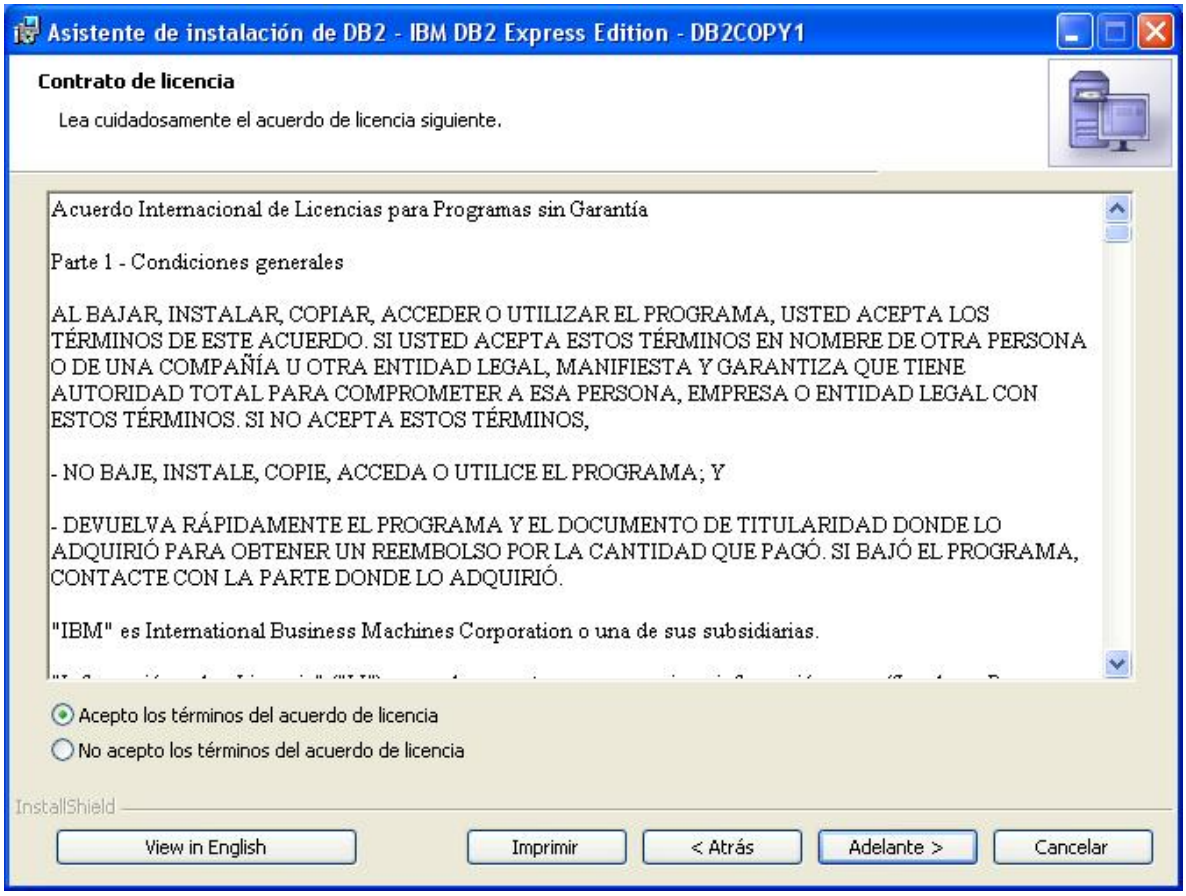


Figura B.3 "Contrato de Licencia del Software".

En la figura B.4, se pueden observar los tres tipos de instalación que posee el sistema, la cual instalará diversas funcionalidades dependiendo de la opción seleccionada.

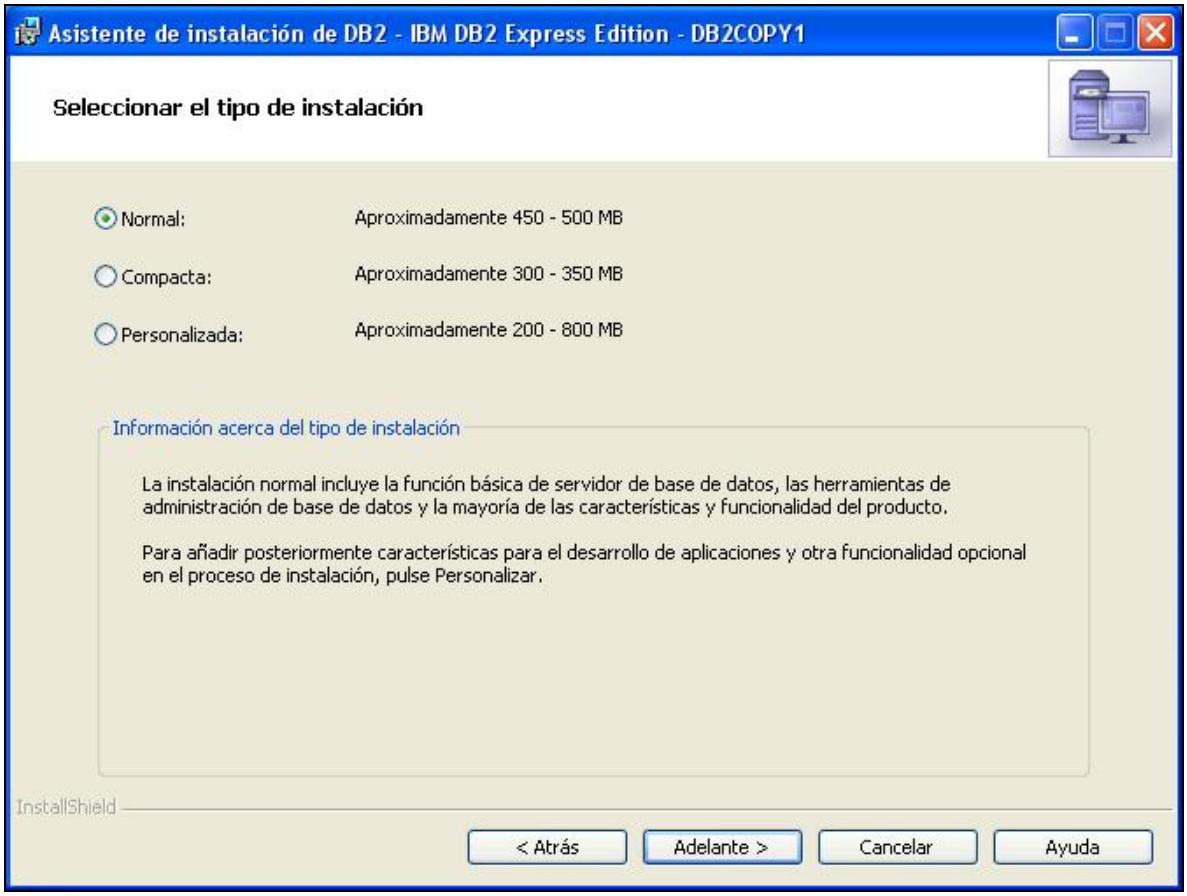


Figura B.4 "Tipos de Instalación del Producto".

Posteriormente, el software ofrece la opción de instalar sólo el software, un archivo de respuestas el cual sirve para evaluar el comportamiento del mismo, o la opción de instalar ambos, como se ve en la figura B.5.

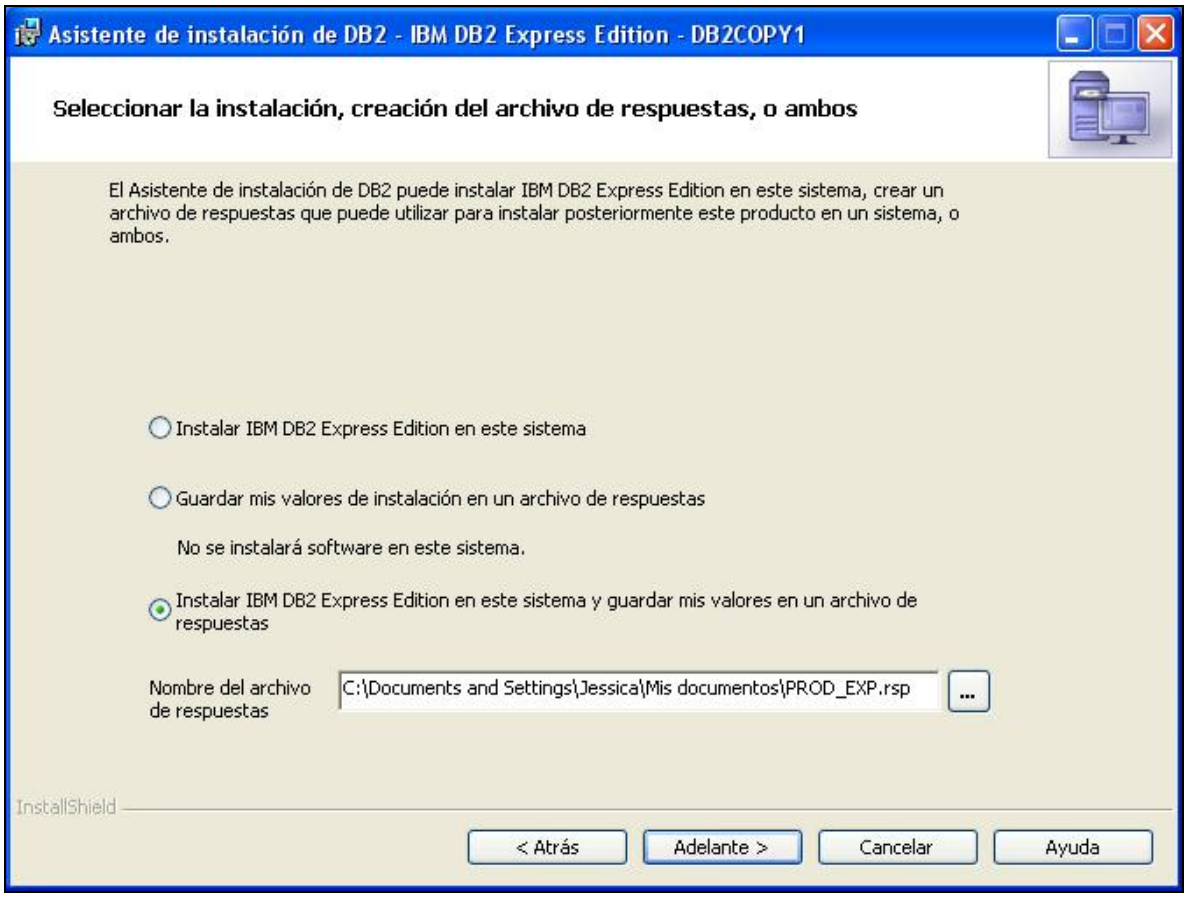


Figura B.5 "Configuración de Instalación".

Luego, se debe seleccionar la carpeta de destino donde se instalarán los archivos del software, apareciendo por defecto la partición del sistema operativo, como se ve en la figura B.6.

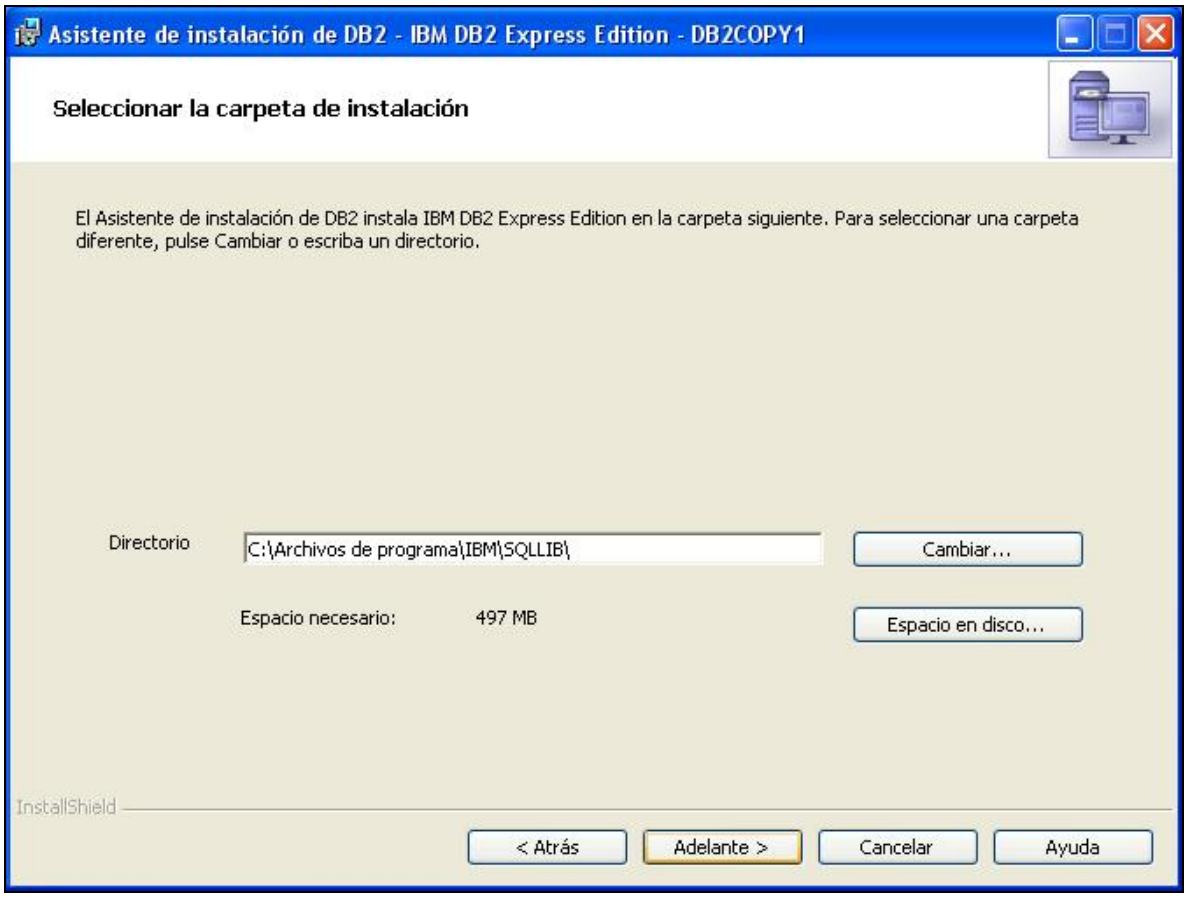


Figura B.6 "Selección Carpeta de Instalación".

En la siguiente pantalla del software que se muestra en la figura B.7, se piden los datos del usuario con el cual se asociará el sistema DB2. Se señala que el usuario se crea automáticamente, y con permiso de administrador del equipo.

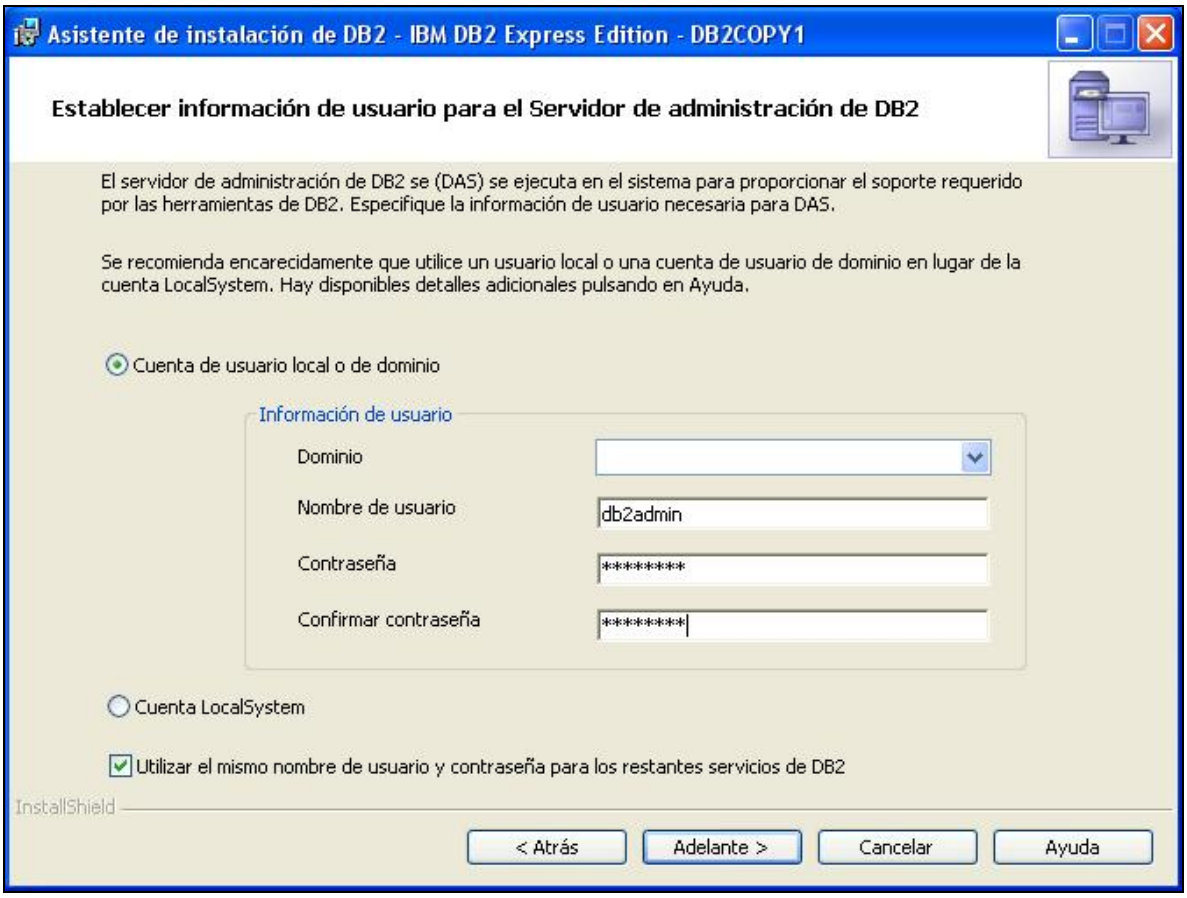


Figura B.7 "Información de Usuario".

En la figura B.8, se muestra la creación de instancias necesarias para la instalación de DB2.

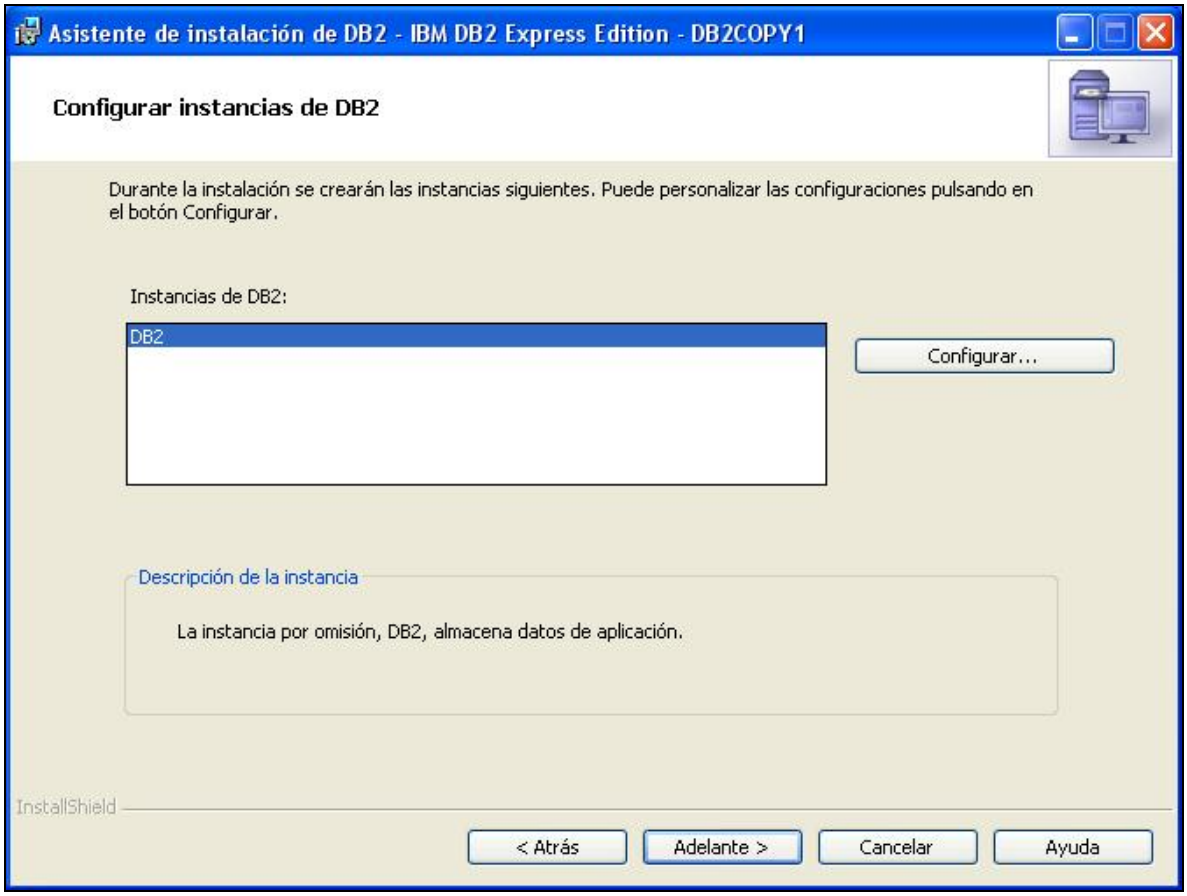


Figura B.8 "Configuración de Instancias".

Antes de comenzar la instalación de los archivos por parte del asistente, se despliega un resumen con la información ingresada por parte del usuario, otorgando la opción de regresar para modificar alguno de ellos, salir del asistente, o confirmar los datos para proceder a la instalación de los archivos, como se ve en la figura B.9.

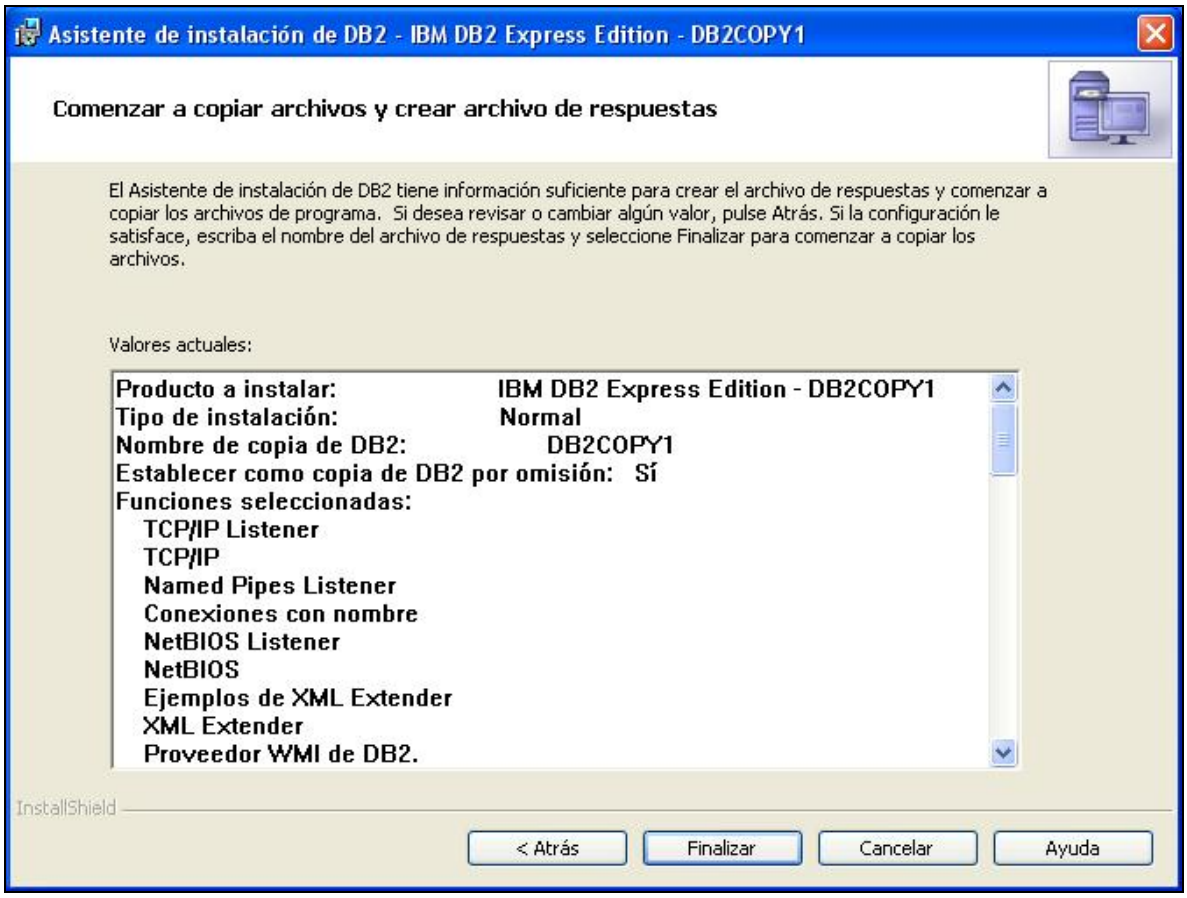


Figura B.9 "Creación de Archivos".

Si se confirmaron todos los datos descritos anteriormente, se procede a la instalación de los archivos del software, en donde se puede apreciar una pantalla como la que se muestra en la figura B.10.

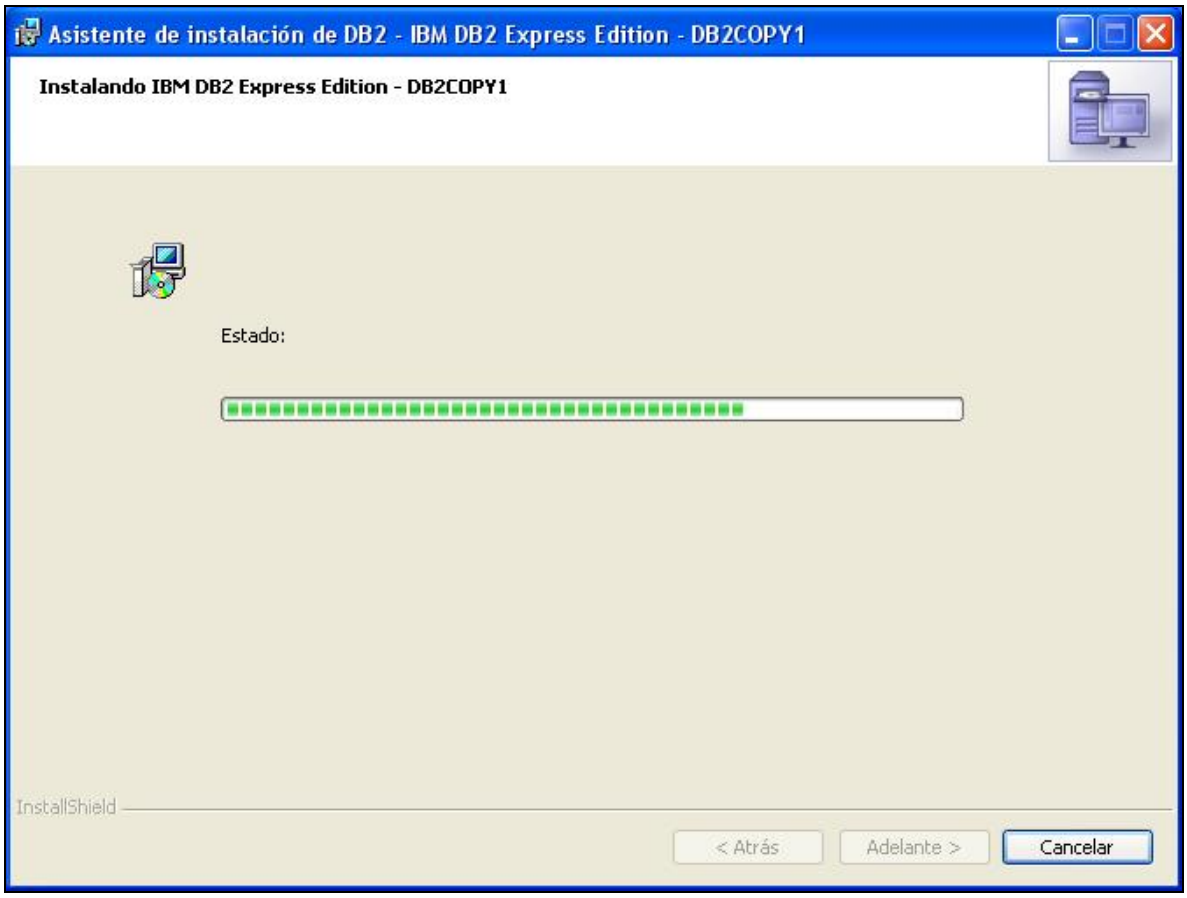


Figura B.10 "Proceso de Instalación".

Posterior a la instalación de los archivos, se muestra otra ventana con el mensaje de finalización, en donde se informa al usuario si se realizó de forma satisfactoria, o existió algún tipo de problema durante la instalación, como se ve en la figura B.11.

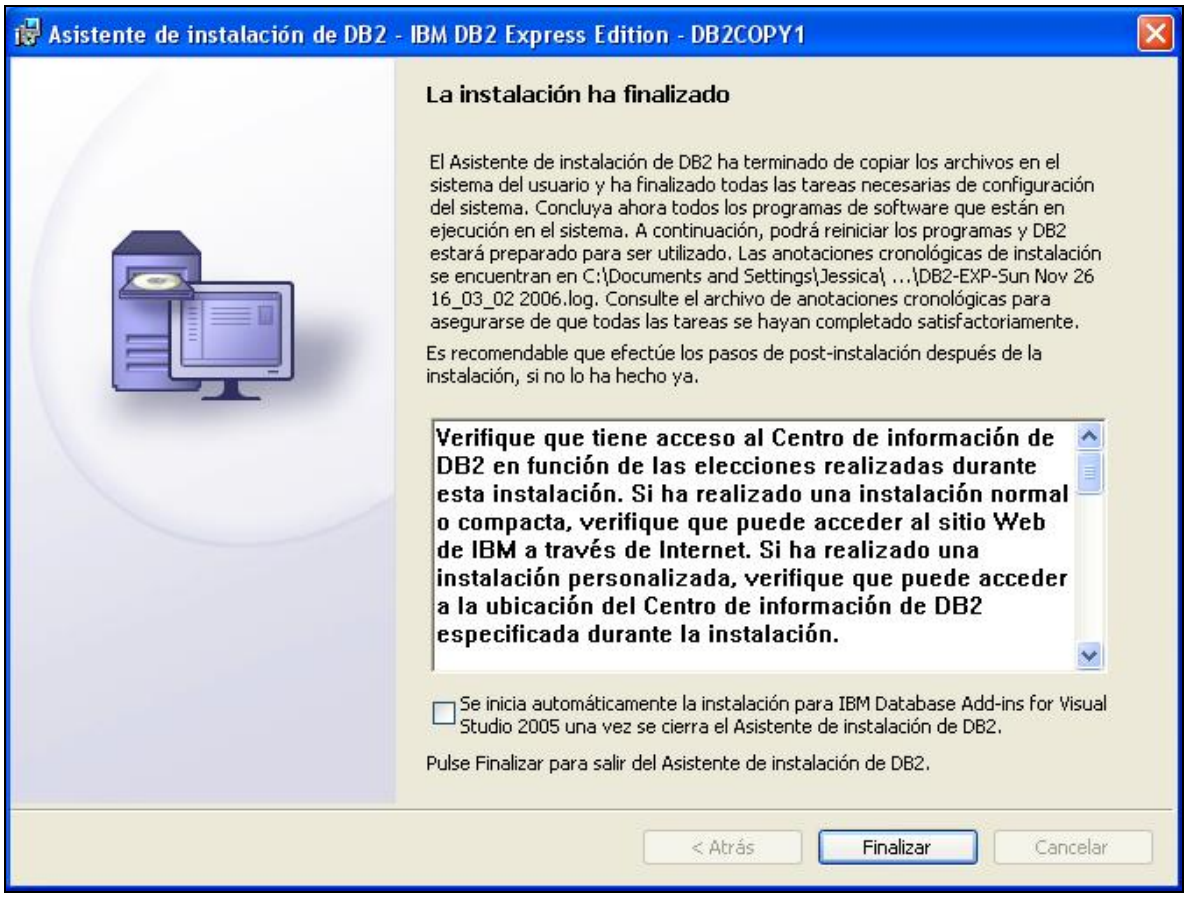


Figura B.11 "Fin de la Instalación".

Si todo el proceso se realizó correctamente, y luego de finalizar la instalación se inicializan los servicios automáticamente, se despliega un asistente con los primeros pasos del Software de Gestión de Información como se ve en la figura B.12.

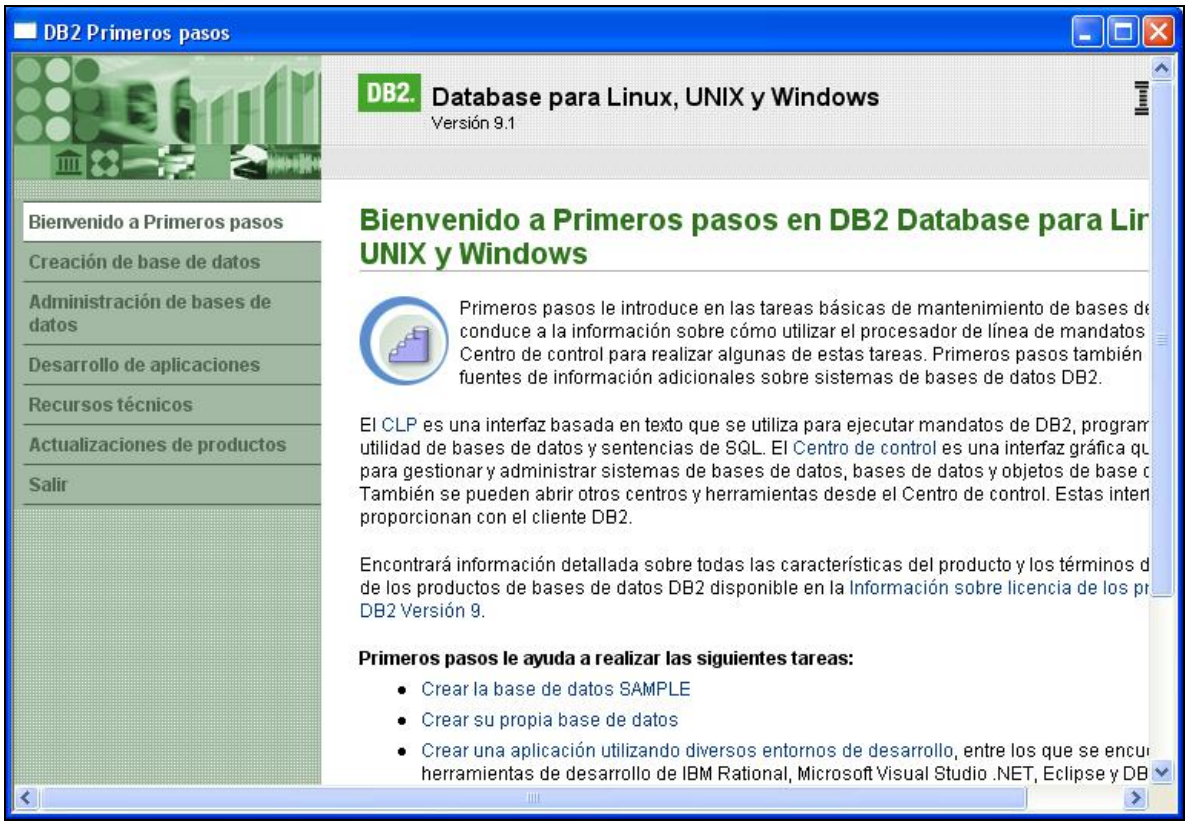


Figura B.12 "Primeros pasos en DB2".

ANEXO C

“Procedimientos de Carga y Mantenimiento del Almacén de Datos”

C. PROCEDIMIENTOS DE POBLAMIENTO, CARGA Y MANTENIMIENTO DEL ALMACÉN DE DATOS.

C.1. POBLAMIENTO DEL ALMACÉN DE DATOS.

Los siguientes procedimientos realizan el proceso de Poblamiento del Almacén de Datos, estos procedimientos se implementaron en el módulo de Procedimientos Almacenados con el que cuenta el administrador de Base de Datos DB2.

C.1.1. Carga la tabla dim_tiempo.

```
CREATE PROCEDURE JESSICA.DIM_TIEMPO (  )
P1: BEGIN
    declare va integer;
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select distinct id_semestre
            from jessica.ingreso ;
        DECLARE cursor2 CURSOR FOR
            select distinct count(id_semestre)-1
               from jessica.ingreso ;
    OPEN cursor1;
    OPEN cursor2;
    fetch from cursor2 into contador;
    fetch from cursor1 into va;
    while (contador>=1 ) do
       INSERT INTO JESSICA.DIM_TIEMPO VALUES (contador,va,va);
       fetch from cursor1 into va;
       set contador = contador - 1;
    end while;
END P1
```

C.1.2. Carga la tabla dim_ciudad.

```
CREATE PROCEDURE JESSICA.DIM_CIUDAD (  )
P1: BEGIN
    declare va integer;
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select distinct ciudad
            from jessica.alumno;
        DECLARE cursor2 CURSOR FOR
            select distinct count(id_semestre)-1
              from jessica.ingreso ;
    OPEN cursor1;
    OPEN cursor2;
    fetch from cursor2 into contador;
    fetch from cursor1 into va;
    while (contador>=1 ) do
       INSERT INTO JESSICA.DIM_CIUDAD VALUES (contador,va,va);
       fetch from cursor1 into va;
       set contador = contador - 1;
    end while;
END P1
```

C.1.3. Carga la tabla dim_asignatura.

```
CREATE PROCEDURE JESSICA.DIM_ASIGNATURA (  )
P1: BEGIN
insert into jessica.dim_asignatura values('1','Electromagnetismo');
insert into jessica.dim_asignatura values('2','Mecánica');
insert into jessica.dim_asignatura values('3','Sistemas de Información');
insert into jessica.dim_asignatura values('4','Algebra');
insert into jessica.dim_asignatura values('5','Base de Datos');
insert into jessica.dim_asignatura values('6','Automatas');
insert into jessica.dim_asignatura values('7','Gestión');
```

```
insert into jessica.dim_asignatura values('8','Circuitos');
END P1
```

C.1.4. Carga la tabla dim_genero.

```
CREATE PROCEDURE JESSICA.DIM_GENERO (  )
P1: BEGIN
insert into jessica.dim_genero values('1','Masculino');
insert into jessica.dim_genero values('2','Femenino');
END P1
```

C.1.5. Carga la tabla dim_region.

```
CREATE PROCEDURE JESSICA.DIM_REGION (  )
P1: BEGIN
insert into jessica.dim_region values('1','Primera Región');
insert into jessica.dim_region values('2','Segunda Región');
insert into jessica.dim_region values('3','Tercera Región');
insert into jessica.dim_region values('4','Cuarta Región');
insert into jessica.dim_region values('5','Quinta Región');
insert into jessica.dim_region values('6','Sexta Región');
insert into jessica.dim_region values('7','Séptima Región');
insert into jessica.dim_region values('8','Octava Región');
insert into jessica.dim_region values('9','Novena Región');
insert into jessica.dim_region values('10','Décima Región');
insert into jessica.dim_region values('11','Decimoprimera Región');
insert into jessica.dim_region values('12','decimosegunda Región');
END P1
```

C.1.6. Carga la tabla fact_duracion

```
CREATE PROCEDURE JESSICA.FACT_DURACION (  )
P1: BEGIN
    declare v1 integer;
    declare v2 varchar(10);
    declare v3 integer;
    declare v4 integer;
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select id_tiempo,a.matricula,b.id_semestre-a.id_semestre,b.id_semestre
            from jessica.dim_tiempo,jessica.ingreso a left join jessica.titulacion b on
            a.matricula=b.matricula
            where a.id_semestre=id_sem  ;
    DECLARE cursor2 CURSOR FOR
            select count(id_tiempo)
            from jessica.dim_tiempo,jessica.ingreso a left join jessica.titulacion b on
            a.matricula=b.matricula
            where a.id_semestre=id_sem  ;

   OPEN cursor1;
   OPEN cursor2;
   fetch from cursor2 into contador;
   fetch from cursor1 into v1,v2,v3,v4;
   while (contador>=1 ) do
      INSERT INTO JESSICA.fact_duracion VALUES (v2,v1,v3,v4);
      fetch from cursor1 into v1,v2,v3,v4;
      set contador = contador - 1;
   end while;
END P1
```

C.1.7. Carga la tabla Fact_alumnas.

```
CREATE PROCEDURE JESSICA.FACT_ALUMNAS (  )
P1: BEGIN
      declare v1 varchar(1);
      declare v2 integer;
      declare v3 varchar(10);
      declare contador integer;
      DECLARE cursor1 CURSOR FOR
            select a.id_genero,b.id_tiempo,c.matricula
            from jessica.dim_genero a,jessica.dim_tiempo b,jessica.alumno
            c,jessica.ingreso d
            where a.genero=c.genero and b.id_sem=d.id_semestre and
            c.matricula=d.matricula ;
      DECLARE cursor2 CURSOR FOR
            select count(a.id_genero)
            from jessica.dim_genero a,jessica.dim_tiempo b,jessica.alumno
            c,jessica.ingreso d
            where a.genero=c.genero and b.id_sem=d.id_semestre and
            c.matricula=d.matricula  ;
      OPEN cursor1;
      OPEN cursor2;
       fetch from cursor2 into contador;
       fetch from cursor1 into v1,v2,v3;
      while (contador>=1 ) do
          INSERT INTO JESSICA.fact_alumnas VALUES (v1,v2,v3);
          fetch from cursor1 into v1,v2,v3;
          set contador = contador - 1;
       end while;
END P1
```

C.1.8. Carga la tabla Fact_cant_causales.

```
CREATE PROCEDURE JESSICA.FACT_CANT_CAUSALES (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare v4 varchar(5);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
                select c.id_genero,b.matricula,semestre,articulo
           from jessica.causal a,jessica.alumno b,jessica.dim_genero c
           where c.genero=b.genero and a.matricula=b.matricula      ;
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula) from jessica.causal a,jessica.alumno
           b,jessica.dim_genero c
           where c.genero=b.genero and a.matricula=b.matricula;
    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3,v4;
     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_cant_causales VALUES (v1,v2,v3,v4);
        fetch from cursor1 into v1,v2,v3,v4;
        set contador = contador - 1;
    end while;
END P1
```

C.1.9. Carga la tabla Fact_alumnos_region

```
CREATE PROCEDURE JESSICA.FACT_ALUMNOS_REGION (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select a.id_region,b.id_ciudad,c.matricula
            from jessica.dim_region a,jessica.dim_ciudad b,jessica.alumno c
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula)
          from jessica.dim_region a,jessica.dim_ciudad b,jessica.alumno c

    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3;
     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_alumnos_region VALUES (v1,v2,v3);
        fetch from cursor1 into v1,v2,v3;
        set contador = contador - 1;
    end while;
END P1
```

C.1.10. Carga la tabla Fact_aprobacion2

```
CREATE PROCEDURE JESSICA.FACT_APROBACION2 (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select a.id_asignatura,b.nota,c.matricula
            from jessica.dim_asignatura a,jessica.alumno_ramo b,jessica.alumno c
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula)
          from jessica.dim_asignatura a,jessica.alumno_ramo b,jessica.alumno c

    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3;
     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_aprobacion2 VALUES (v1,v2,v3);
        fetch from cursor1 into v1,v2,v3;
        set contador = contador - 1;
    end while;
END P1
```

C.2. CARGA Y MANTENIMIENTO DEL ALMACÉN DE DATOS.

Los siguientes procedimientos realizan el proceso de Carga y mantenimiento del Almacén de Datos, estos procedimientos se implementaron en el módulo de Procedimientos Almacenados con el que cuenta el administrador de Base de Datos DB2.

C.2.1.

```
CREATE PROCEDURE JESSICA.CARGA_DIM_TIEMPO (  )
P1: BEGIN
     declare va integer;
     declare contador integer;
     DECLARE cursor1 CURSOR FOR
               select distinct id_semestre
               from jessica.ingreso ;
     DECLARE cursor2 CURSOR FOR
              select distinct count(id_semestre)-1
              from jessica.ingreso ;
    OPEN cursor1;
    OPEN cursor2;
    fetch from cursor2 into contador;
    fetch from cursor1 into va;
    delete from jessica.dim_tiempo;
    while (contador>=1 ) do
    INSERT INTO JESSICA.DIM_TIEMPO VALUES (contador,va,va);
    fetch from cursor1 into va;
    set contador = contador - 1;
   end while;
END P1
```

C.2.2.

```
CREATE PROCEDURE JESSICA.DIM_CIUDAD (  )
P1: BEGIN
    declare va integer;
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select distinct ciudad
            from jessica.alumno;
       DECLARE cursor2 CURSOR FOR
            select distinct count(id_semestre)-1
              from jessica.ingreso ;
    OPEN cursor1;
    OPEN cursor2;
    fetch from cursor2 into contador;
    fetch from cursor1 into va;
    delete from jessica.dim_ciudad;
    while (contador>=1 ) do
      INSERT INTO JESSICA.DIM_CIUDAD VALUES (contador,va,va);
      fetch from cursor1 into va;
      set contador = contador - 1;
    end while;
END P1
```

C.2.3.

```
CREATE PROCEDURE JESSICA.CARGA_FACT_DURACION (  )
P1: BEGIN
    declare v1 integer;
    declare v2 varchar(10);
    declare v3 integer;
    declare v4 integer;
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select id_tiempo,a.matricula,b.id_semestre-a.id_semestre,b.id_semestre
            from jessica.dim_tiempo,jessica.ingreso a left join jessica.titulacion b on
            a.matricula=b.matricula
            where a.id_semestre=id_sem  ;
    DECLARE cursor2 CURSOR FOR
            select count(id_tiempo)
            from jessica.dim_tiempo,jessica.ingreso a left join jessica.titulacion b on
            a.matricula=b.matricula
            where a.id_semestre=id_sem  ;

   OPEN cursor1;
   OPEN cursor2;
   fetch from cursor2 into contador;
   fetch from cursor1 into v1,v2,v3,v4;
   delete from jessica.fact_duracion;
   while (contador>=1 ) do
      INSERT INTO JESSICA.fact_duracion VALUES (v2,v1,v3,v4);
      fetch from cursor1 into v1,v2,v3,v4;
      set contador = contador - 1;
   end while;
END P1
```

C.2.4

```
CREATE PROCEDURE JESSICA.CARGA_FACT_ALUMNAS (  )
P1: BEGIN
      declare v1 varchar(1);
      declare v2 integer;
      declare v3 varchar(10);
      declare contador integer;
      DECLARE cursor1 CURSOR FOR
            select a.id_genero,b.id_tiempo,c.matricula
            from jessica.dim_genero a,jessica.dim_tiempo b,jessica.alumno
            c,jessica.ingreso d
            where a.genero=c.genero and b.id_sem=d.id_semestre and
            c.matricula=d.matricula ;
      DECLARE cursor2 CURSOR FOR
            select count(a.id_genero)
            from jessica.dim_genero a,jessica.dim_tiempo b,jessica.alumno
            c,jessica.ingreso d
            where a.genero=c.genero and b.id_sem=d.id_semestre and
            c.matricula=d.matricula  ;
      OPEN cursor1;
      OPEN cursor2;
       fetch from cursor2 into contador;
       fetch from cursor1 into v1,v2,v3;
       delete from jessica.fact_alumnas;
      while (contador>=1 ) do
          INSERT INTO JESSICA.fact_alumnas VALUES (v1,v2,v3);
          fetch from cursor1 into v1,v2,v3;
          set contador = contador - 1;
       end while;
END P1
```

C.2.5.

```
CREATE PROCEDURE JESSICA.CARGA_FACT_CANT_CAUSALES (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare v4 varchar(5);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
                select c.id_genero,b.matricula,semestre,articulo
           from jessica.causal a,jessica.alumno b,jessica.dim_genero c
           where c.genero=b.genero and a.matricula=b.matricula      ;
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula) from jessica.causal a,jessica.alumno
           b,jessica.dim_genero c
           where c.genero=b.genero and a.matricula=b.matricula;
    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3,v4;
     delete from jessica.fact_cant_causales;

     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_cant_causales VALUES (v1,v2,v3,v4);
        fetch from cursor1 into v1,v2,v3,v4;
        set contador = contador - 1;
    end while;
END P1
```

C.2.6.

```
CREATE PROCEDURE JESSICA.FACT_ALUMNOS_REGION (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select a.id_region,b.id_ciudad,c.matricula
            from jessica.dim_region a,jessica.dim_ciudad b,jessica.alumno c
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula)
          from jessica.dim_region a,jessica.dim_ciudad b,jessica.alumno c

    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3;
     delete from jessica.fact_alumnos_region;
     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_alumnos_region VALUES (v1,v2,v3);
        fetch from cursor1 into v1,v2,v3;
        set contador = contador - 1;
    end while;
END P1
```

C.1.7.

```
CREATE PROCEDURE JESSICA.FACT_APROBACION2 (  )
P1: BEGIN
    declare v1 varchar(1);
    declare v2 varchar(10);
    declare v3 varchar(15);
    declare contador integer;
    DECLARE cursor1 CURSOR FOR
            select a.id_asignatura,b.nota,c.matricula
            from jessica.dim_asignatura a,jessica.alumno_ramo b,jessica.alumno c
    DECLARE cursor2 CURSOR FOR
           select count(b.matricula)
          from jessica.dim_asignatura a,jessica.alumno_ramo b,jessica.alumno c

    OPEN cursor1;
    OPEN cursor2;
     fetch from cursor2 into contador;
     fetch from cursor1 into v1,v2,v3;
     delete from jessica.fact_aprobacion2;
     while (contador>=1 ) do
        INSERT INTO JESSICA.fact_aprobacion2 VALUES (v1,v2,v3);
        fetch from cursor1 into v1,v2,v3;
        set contador = contador - 1;
    end while;
END P1
```