COLEGIO DE POSTGRADUADOS

INSTITUCIÓN DE ENSEÑANZA E INVESTIGACIÓN EN CIENCIAS AGRÍCOLAS

CAMPUS MONTECILLO

POSTGRADO DE SOCIOECONOMÍA, ESTADÍSTICA E INFORMÁTICA

CÓMPUTO APLICADO

COMPARACIÓN DE UNA CONSULTA TÍPICA DE ANÁLISIS DE NEGOCIOS USANDO DOS ESTRUCTURAS DIFERENTES DE BASES DE DATOS

JUAN GONZÁLEZ ESPINOSA

T E S I S

PRESENTADA COMO REQUISITO PARCIAL

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS

MONTECILLO, TEXCOCO, EDO. DE MÉXICO

2011

La presente tesis, titulada: **"COMPARACIÓN DE UNA CONSULTA TÍPICA DE ANÁLISIS DE NEGOCIOS USANDO DOS ESTRUCTURAS DIFERENTES DE BASES DE DATOS",** realizada por el alumno: **Juan González Espinosa,** bajo la dirección del Consejo Particular indicado, ha sido aprobada por el mismo y aceptada como requisito parcial para obtener el grado de:

MAESTRO EN CIENCIAS

SOCIOECONOMÍA, ESTADÍSTICA E INFORMÁTICA

CÓMPUTO APLICADO

CONSEJO PARTICULAR

CONSEJERO: _____

Dr. Juan R. Bauer Mengelberg

ASESOR: _____

M.C. Margarita Cruz Millan

ASESOR: _____

Dr. David Hebert del Valle Paniagua

Montecillo, Texcoco, Edo. de México, Abril de 2011.

DEDICATORIAS

Dedico esta tesis a todos los que creyeron en mi, a toda la gente que me apoyo, a mis amigos y familiares y ha esta institución que me ha formado, pero en especial se la dedico a mis padres y esposa, a quienes agradezco de todo corazón por su cariño, comprensión y consejos. En todo momento los llevo conmigo.

AGRADECIMIENTOS

PROFESORES DEL PROGRAMA
Por haberme ayudado con su conocimiento en esta etapa de mi formación profesional.

AL CONSEJO NACIONAL DE CIENCIA Y TECNOLOGÍA (CONACYT)
Por el apoyo económico que me otorgó para poder llevar a cabo mis estudios de Maestría.

AL COLEGIO DE POSTGRADUADOS
En especial al Programa de Cómputo Aplicado, por todas las enseñanzas y las experiencias obtenidas y por la amistad recibida de todos los que en él laboran.

A MIS COMPAÑEROS DEL COLEGIO DE POSTGRADUADOS
Por todos los momentos que compartimos a lo largo de esta etapa.

Un agradecimiento muy especial a mi consejero Dr. Juan R. Bauer Mengelberg por compartir conmigo sus conocimientos, por brindarme sabios consejos, por su tiempo y dedicación y sobre todo por su paciencia le estoy muy agradecido.

CONTENIDO

Página

DEDICATORIAS i
AGRADECIMIENTOS ii
ÍNDICE DE FIGURAS vii
ÍNDICE DE TABLAS viii
RESUMEN ix
ABSTRACT x
CAPÍTULO 1. INTRODUCCIÓN 1
CAPÍTULO 2. LAS BODEGAS DE DATOS Y EL ANÁLISIS DE NEGOCIOS 7
Introducción 7
2.1 Los usos típicos de las bodegas de datos 8
2.1.1 Inteligencia de Negocios 8
2.1.3 Análisis de Negocios 9
2.1.4 Lo que tienen en común 10
2.2 Características, atributos y usos de Bodegas de Datos 12
2.3 Las bases de datos relacionales 15
2.3.1 introducción 15
2.3.2 Los índices de una base de datos relacional 16
2.4 Árboles B (B-trees) 18
2.5 Índice GIN 20
2.6 El uso de bitmaps 22
2.7 El lenguaje SQL 24
2.8 El modelo de datos y los índices 25
2.9 Plan de ejecución 25
2.10 Un ejemplo de un plan de ejecución 29
2.11 Bases de datos Columnares 30
2.12 Bases Orientadas o objetos e hibridas 31
2.13 Bases de datos con el concepto de Key-value 31
2. 14 Tecnologías tipo Appliance Technology 32
2.15 Tecnologías que combinan diversas arquitecturas 34
2.16 El DBB 34
CAPÍTULO 3. DESCRIPCIÓN DEL DBB, UN MODELO HÍBRIDO 35
3.1 Antecedentes del DBB 35
3.2 los componentes del DBB 38
3.3. Descripción panorámica del DBB 39
3.4 Los contextos, las marcas y el KBC 40
3.4.1 Los contextos 40
3.4.2 Las marcas 41
3.5 Los campos de una UBI 42
3.5.1 Los campos fijos de una UBI 43
3.5.2 Campos adicionales 44
3.5.3 Los textos 45
3.5.4 Las marcas individuales 45
3.5.5 Objetos asociados a la ficha 46
3.5.6 Apuntadores (Punteros) y tamaños 46

3.5.7 Campos de Auditoria 46
3.6 Clases 47
3.6.1 Especificación de ciertos atributos de la clase misma 48
3.6.2 Selección de las opciones que usa esa clase (entre las disponibles) 48
3.6.3 Uso de los campos fijos de una UBI 48
3.6.4 Indicación de cuáles de los textos se usarán 49
3.6.5 Definición de los grupos de campos (si los usa) 49
3.6.6 Especificación de los campos "adicionales" que se usarán 49
3.6.7 Uso de los 16 campos "si o no" 50
3.6.8 Especificación de los tipos de objetos asociados (cuando los usa la clase) 50
3.6.9 Marcado automático de campos 51
3.6.10 Indicación de los contextos que usa la clase 51
3.6.11 Opciones de deformación de fichas 51
3.6.12 Cifras de auditoria de las UBI 51
3.7 Actualización de datos en DBB 52
3.8 Consultas en el DBB 52
3.9 Filtros de listas de resultados 53
3.10 Otras formas de entregar resultados 54
3.11 Cómo se almacenan los datos en DBB 55
3.12 Opciones disponibles para las aplicaciones 57
CAPÍTULO 4. METODOLOGÍA Y HERRAMIENTAS 60
Introducción: El Objetivo de la investigación 60
4.1 La consulta a ventas de libros utilizada para comparar su implementación en las dos tecnologías 60
4.2 Etapas del estudio 62
4.3 Materiales y tecnologías utilizadas 63
4.3.1 Lenguaje de programación 63
4.3.2 Manejador de bases de datos relacionales (RDBMS) 64
4.3.3 Dispositivos de almacenamiento 65
4.3.4 Computadoras utilizadas 65
CAPÍTULO 5. IMPLEMENTACIÓN DE LA SITUACIÓN DE NEGOCIOS EN UNA BASE DE DATOS RELACIONAL 66
Introducción 66
5.1 Las tablas de la base de datos 66
5.2 Descripción de los campos de las tablas de la base de datos 67
CAPÍTULO 6. LA GENERACIÓN DE DATOS SIMULADOS 70
6.1 Atributos deseados de los datos generados 70
6.2 Generación de los libros con sus atributos 70
6.2.1 Descripción general 70
6.2.2 Las tablas creadas para los catálogos 71
6.2.3 Resumen del programa con el que se generaron los libros 74
6.3 Generación de los clientes con sus atributos 76
6.3.1 Descripción general 76
6.3.2 El programa para generar los clientes 77

6.4 Generación de las ventas de libros a clientes 78
6.4.1 Descripción general 78
6.4.2 Resumen del programa que generó las ventas 80
6.4.3 Comentarios sobre el proceso de generación de datos 81
CAPÍTULO 7. LAS CONSULTAS EN LA BASE DE DATOS RELACIONAL 82
Introducción 82
7.1 Las consultas 82
7,2 Las sentencias SQL 82
7.3 Rutinas con las que el programa arma la sentencia para un libro y un cliente 84
7.4 Conceptos básicos de índices de una base de datos relacional 85
7.5 Las consultas efectuadas y comentarios sobre las mismas 87
7.6 El programa con el que se ejecutaron las consultas 89
7.7 Los tiempos de respuesta obtenidos 90
7.10 Comentarios acerca de las duraciones en diversas bases 94
7. 11 Impacto de la presencia de índices en los planes de ejecución 95
7.11.1 Usa Base # 1, consulta básica sin condiciones adicionales 98
7.11.2 Usa Base # 4, consulta básica sin condiciones adicionales 98
7.11.3 Usa Base # 1, sin "*" en Select de LIBROS 99
7.11.3 Usa Base # 2, consulta básica sin condiciones adicionales 99
7.11.4 Usa Base # 5, consulta básica sin condiciones adicionales 100
7.11.5 Usa Base # 1, Excluye clientes de categoría 6 100
7.11.6 Usa Base # 1, Excluye clientes de categoría 6 y sin el "*" en el Select 101
CAPÍTULO 8. IMPLEMENTACIÓN DE LA SITUACIÓN DE NEGOCIOS EN DBB 102
Introducción 102
8.1 Seleccionar las opciones del DBB que se usará la aplicación 102
8.2 Definir los contextos que usará 107
8.3 Definir las clases de UBI 108
8.3.1 La clase CLIENTES 109
8.3.2 La clase LIBROS 115
8.4 Construir una estructura de directorios para los datos 117
8.5 Preparar el uso del paquete KBC 117
8.6 Carga de datos simulados (ETL) a la estructura definida 119
8.6.1 Diseño del proceso ETL 119
8.6.2 Diseño, programación y pruebas de los programas 120
8.6.3 Creación de fichas de libros 121
8.6.4 Creación de fichas de clientes 123
8.6.5 Actualización del contexto 12 con los clientes que compraron cada libro 124
8.6.6 Actualización de los vectores paralelos de clientes 126
8.6.7 Analizar los tiempos de carga de datos 127
8.6.8 Modificar los programas de carga si fuera necesario 127
CAPÍTULO 9. LAS CONSULTAS CON DBB 128

9.1 .....**La consulta básica** 128
9.2 Duraciones obtenidas al efectuar las consultas en DBB..... 130
9.3 Consultas con criterios de selección adicionales..... 131
CAPÍTULO 10. COMPARACIONES ENTRE LAS 2 TECNOLOGÍAS..... 132
Introducción..... 132
Espacio en disco 132
Comparación de las duraciones de las consultas en ambas tecnologías. 133
Conclusiones de estas comparaciones 135
CAPÍTULO 11...... **EFECTO DE LA DUPLICACIÓN DEL TAMAÑO DEL ACERVO** 136
Introducción..... 136
Tiempos de respuesta a consultas..... 137
Conclusiones de estas comparaciones 138
CAPÍTULO 12. OTRAS CONSULTAS CONTEMPLADAS EN SOB 139
12.1 Introducción..... 139
12.2 Las condiciones que se probaron..... 139
12.2.1 Sólo ofrecer libros del mismo tema que L1 140
12.2.3 Sólo libros que están disponibles en uno de los idiomas en los que se ofrece L1..... 142
12.2.4 Sólo contemplar clientes que compraron L1 en los últimos M meses ……...144
12.2.5 Sólo se ofrecen libros que compraron en los últimos M meses los clientes que compraron L1 144
12.2.6 Sólo incluir clientes que compraron por lo menos otro libro del mismo tema que L1...... 145
12.2.7 Sólo tomar en cuenta clientes que residen en el mismo país o ciudad que C1 (el que dio lugar a la consulta.)..... 146
CONCLUSIONES..... 147
REFERENCIAS 148

ÍNDICE DE FIGURAS

Página

FIGURA 1. UN ÁRBOL B+ 19
FIGURA 2. UN ÍNDICE GIN 21
FIGURA 3. EJEMPLO DE MARCADO DE PALABRAS 22
FIGURA 4. UN BITMAP 23
FIGURA 5. EJEMPLO DE DOS BITMAPS 23
FIGURA 6. PLAN DE EJECUCIÓN DE LA CONSULTA BÁSICA DEL SOB 30
FIGURA 7. LA AGRUPACIÓN DE CAMPOS DE UNA UBI 43
FIGURA 8. OPCIONES RELACIONADAS CON LA FUNCIONALIDAD GENERAL (DBB) 57
FIGURA 9. LAS TABLAS DE LA BASE DE DATOS 66
FIGURA 10. COMPARACIÓN DEL USO DE EXCEPT E INTERSECT 84
FIGURA 11. PANTALLA PARA INDICAR PARÁMETROS Y FORMA DE EJECUCIÓN DE CONSULTAS. 89
FIGURA 12. LA FORMA EN LA QUE SE MUESTRAN LOS RESULTADOS 90
FIGURA 13. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #1 98
FIGURA 14. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #4 98
FIGURA 15. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #1 SIN EL SELECT “*” 99
FIGURA 16. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #2 99
FIGURA 17. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #5 100
FIGURA 18. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #5 EXCLUYENDO CLIENTES DE LA CATEGORÍA 6 100
FIGURA 19. PLAN DE EJECUCIÓN DE LA CONSULTA BASICA CON BASE #1 101
FIGURA 20. LA FORMA PARA CREAR UNA CLASE 111
FIGURA 21. LA DEFINICIÓN DE LA CLASE # 2: CLIENTE 112
FIGURA 22. CAMBIOS AL USO DE UN CAMPO FIJO DE LA CLASE 113
FIGURA 23. CAMBIOS AL USO DE LOS 16 CAMPOS SI-O-NO DE LA CLASE. 114
FIGURA 24. FORMA PARA INDICAR EL TIPO DE MATERIALES ASOCIADOS DE UNA CLASE 114

ÍNDICE DE TABLAS

Página

TABLA 1. ATRIBUTOS DE UN CONTEXTO 40
TABLA 2. LOS CAMPOS FIJOS DE UNA UBI 44
TABLA 3. LAS OPCIONES DEL PAQUETE DBB 58
TABLA 4. OPCIONES RELACIONADAS CON UNA INSTANCIA 58
TABLA 5. CAMPOS DE LA TABLA LIBROS DE LA BASE DE DATOS 68
TABLA 6. CAMPOS DE LA TABLA CLIENTES DE LA BASE DE DATOS 68
TABLA 7. CAMPOS DE LA TABLA VENTAS DE LA BASE DE DATOS 69
TABLA 8. TABLAS ADICIONALES DE LA BASE DE DATOS 69
TABLA 9. LOS TIPOS DE LIBRO Y LOS VALORES ASOCIADOS 72
TABLA 10. NÚMERO DE LIBROS POR TIPO DE LIBRO Y TEMA 73
TABLA 11. LA TABLA CATEGORÍAS DE CLIENTES 76
TABLA 12. DISTRIBUCIÓN DEL NÚMERO DE VENTAS PARA LIBROS DEL TIPO 1 79
TABLA 13. PORCENTAJES DE VENTAS DE LIBROS DE UN TIPO 79
TABLA 14. LAS BASES DE DATOS CREADAS PARA DETERMINAR EL IMPACTO DE 86
TABLA 15. LIBROS SELECCIONADOS PARA LAS CONSULTAS 88
TABLA 16. DURACIONES DE LAS CONSULTAS EFECTUADAS 91
TABLA 17. RESUMEN DE CONSULTAS EN LAS 7 BASES (SÓLO DURACIONES PROMEDIO) 92
TABLA 18. DURACIONES EN SEGUNDOS Y FRACCIÓN DE LA CONSULTA BÁSICA 93
TABLA 19. DURACIONES PROMEDIO EN SEGUNDOS Y FRACCIÓN DE LA CONSULTA BÁSICA 94
TABLA 20. CONSULTA BÁSICA (INCLUYENDO CLIENTE DE LA CATEGORÍA 6) 95
TABLA 21. CONSULTA BÁSICA (EXCLUYENDO CLIENTE DE LA CATEGORÍA 6) 95
TABLA 22. ÍNDICES DE LA TABLA VENTAS DE LAS BASES DE DATOS 96
TABLA 23. CONSULTAS PARA LAS CUALES SE MUESTRAN LOS PLANES DE EJECUCIÓN 96
TABLA 24. LAS OPCIONES DEL PAQUETE DBB QUE SE IMPLEMENTARON 103
TABLA 25. OPCIONES DE LA INSTANCIA SOB 104
TABLA 26. OPCIONES RELACIONADAS CON LAS CLASES 105
TABLA 27.LOS CONTEXTOS DEFINIDOS PARA LA APLICACIÓN SOB 108
TABLA 28. LOS CAMPOS DE UNA UBI DE LA CLASE CLIENTE 110
TABLA 30. SUBDIRECTORIOS DEL DIRECTORIO EL SOB Y LO QUE CONTIENEN 117
TABLA 31. ILUSTRACIÓN DEL PROCESO DE UNIÓN DE LIBROS ADQUIRIDOS POR LOS CLIENTES 130
TABLA 32. DURACIONES EN SEGUNDOS Y FRACCIÓN DE LA CONSULTA BÁSICA 130
TABLA 33. RESUMEN DE CONSULTAS INCLUYENDO LOS CLIENTES DE CATEGORÍA 6 131
TABLA 34. RESUMEN DE CONSULTAS EXCLUYENDO LOS CLIENTES DE CATEGORÍA 6 131
TABLA 35. ESPACIO EN DISCO OCUPADO POR EL ACERVO DE DATOS 132
TABLA 36. ESPACIO EN DISCO OCUPADO EN AMBAS TECNOLOGÍAS (CON LA BASE #1) 133
TABLA 37. COMPARACIÓN DE TIEMPOS DE RESPUESTA A LA CONSULTA BÁSICA 134
TABLA 38. COMPARACIÓN DE TIEMPOS DE RESPUESTA A LA CONSULTA BÁSICA 135
TABLA 40. COMPARACIÓN DE TIEMPOS DE RESPUESTA PARA EL ACERVO DOBLE 137
TABLA 41. LAS 7 CONSULTAS DETALLADAS 139

COMPARACIÓN DE UNA CONSULTA TÍPICA DE ANÁLISIS DE NEGOCIOS USANDO DOS ESTRUCTURAS DIFERENTES DE BASES DE DATOS

Juan González Espinosa
Colegio de Postgraduados, 2011

RESUMEN

El crecimiento tanto de los tamaños de los almacenes de datos como de su utilización por empresas y organismos para analizar la información ha dado lugar al desarrollo de diversas tecnologías que se plantean como alternativas a las bases de datos relacionales para estructurar y almacenar los datos: Esto se debe a que éstas no proporcionan tiempos de respuesta adecuados en consultas no planeadas o predecibles como las que se presentan en el análisis de negocios o la minería de datos. Las bases relacionales se crearon para el proceso de transacciones, donde se incluyen en el modelo índices para los usos planeados de los datos. La ausencia de ciertos índices puede resultar en la lectura de toda una o varias tablas (*table scans*), lo que impacta en los tiempos de respuesta.

Además de ciertas alternativas que surgieron al mismo tiempo que los modelos relacionales, se han adoptado otras estructuras para almacenar y usar los datos. En este trabajo se describe una situación de negocios simulada, usando tanto un modelo relacional como el modelo DBB, un modelo basado principalmente en pares llave-valor: algunos datos se guardan en campos fijos, mientras que a otros valores se les asigna una etiqueta que indica el significado del dato. Se obtienen subconjuntos de registros que satisfacen ciertos criterios de selección y se efectúan operaciones lógicas entre ellos. Una librería ofrece a los clientes que adquieren un libro la lista de todos los libros que compraron otros clientes que también compraron el libro en cuestión. Se describen las estructuras usadas y el proceso con el que se generaron los datos – simulados – usados para las comparaciones. Puesto que se redujeron los tiempos de respuesta y el espacio

ocupado por los datos, se concluye que el DBB puede ofrecer una mejor solución para algunas bodegas de datos.

Palabras Clave: Bases de datos, almacenes de datos, bases híbridas, comparación en desempeño, análisis de negocios, índices invertidos generalizados.

COMPARISON OF A TYPICAL BUSINESS ANALYTICS QUERY USING TWO DIFFERENT DATABASE STRUCTURES

ABSTRACT

The growth both of the size of data warehouses and their use to analyze the information they contain has given rise to the development of different technologies that constitute alternatives to the relational databases to organize and store the data, since the latter sometimes do not deliver adequate response times in un-planned o unpredictable queries, such as occur in business analytics or data mining. Relational databases were conceived for transaction processing, where the necessary indices are included according to the planned use of the data. The absence of such indices often results in the need to perform table scans of entire tables, and as these grow in size, the associated input-output activities cause response times to be longer than expected. Besides some alternatives that arose almost simultaneously with relational models, new structures are being used to store and use the data. Additionally, different hardware and software combinations have become alternatives for very large data repositories. DBB is a model based essentially on key-value pairs: some data values are stored in fixed fields, but others are identified by a tag. To obtain a subset of records that satisfy certain conditions, these are formulated as logical operations between key-value pairs, which are included for that purpose in generalized inverted indexes. The paper describes a comparison of response times and total disk space resulting from a business analytics situation implemented both as a relational model as well as

using the DBB model. A bookstore offers customers who purchase a particular book the list of all books bought by other clients who also bought that book. The process used to generate simulated data is described. Since both response times and disk space were reduced using DBB, it seems that there are potential uses of this model.

Keywords: Databases, data warehouses, benchmarks, business analytics, hybrid databases, generalized inverted indices.

CAPÍTULO 1. INTRODUCCIÓN

El crecimiento tanto de los tamaños de los almacenes de datos como su utilización por empresas y organismos para analizar la información ha dado lugar al desarrollo de diversas tecnologías que se plantean como alternativas a las bases de datos relacionales para estructurar y almacenar los datos. Cada vez más investigadores y usuarios afirman que no hay una tecnología óptima para las bodegas de datos (data warehouses) y su explotación para la inteligencia y el análisis de negocios o para la minería de datos. En especial, las bases de datos relacionales (BDR) – hasta hace poco con la mayor participación de mercado en este tipo de situaciones – aparentemente no son la mejor solución en muchos casos.

Uno de los argumentos más claros que se presentan es que las BDR se crearon para sistemas que manejan transacciones, en los cuales han tenido una aceptación y éxito aun mayor a la esperada cuando se las introdujo a los mercados. Esto se debe a que, con un diseño adecuado de los modelos, tras un análisis completo de los requisitos del sistema, era posible determinar los diversos usos de los datos e incluir los índices apropiados en el modelo utilizado, para obtener de este modo subconjuntos de registros sin tener que leer toda una tabla.

Sin embargo, en las aplicaciones como las que se mencionaron anteriormente, se presentan usos no previsibles de los datos. La ausencia de ciertos índices puede resultar en la necesidad de leer toda una tabla, o varias de ellas. En otras palabras, el uso de una tecnología diseñada para un propósito, el uso de los datos en transacciones (on-line transaction processing) no implica que sea ideal para otros usos, como son los análisis de negocios. Uno de los investigadores que describió esta situación usó un chiste que originalmente sirvió para otros propósitos. Una persona le pregunta a alguien en una ciudad cómo llegar del punto en que se encuentra a otro, muy lejano. La respuesta que obtuvo fue: ¡para ir allí, yo no hubiera partido de aquí!

El modo de preparar un modelo de datos relacional para los diversos usos es la inclusión de índices, que proporcionan acceso a subconjuntos de registros de

una tabla, en lugar de tener que leer todos ellos. La ausencia de un índice puede hacer que una consulta sea tardada cuando la tabla tiene un número elevado de registros. A pesar de que los sistemas de gestión de BDR (RDBMS) han sufrido constantes mejoras y cada vez aprovechan más los recursos de las computadoras – incluyendo especialmente la memoria – no se pueden evitar ciertos procesos que alargan las consultas.

De ese modo, además de algunas alternativas formuladas casi en forma simultánea al modelo relacional, especialmente ADABAS (adabas, 2010), basado en listas invertidas, han cobrado importancia, en cuanto a las empresas que las utilizan, las bases columnares ofrecidas por varios proveedores, entre las cuales se puede citar a SYBASE (Frank Teklitz, 2000), ORACLE (Richard Strohm, 2008), VERTICA (Vertica Systems Inc., 2010), INFOBRIGHT (Infobright, 2010), aunque esta dista mucho de ser exhaustiva. Los modelos orientados a objetos y las bases híbridas, también están obteniendo una participación de mercado creciente, destacando POSTGRES como una de las bases de datos hibridas de mayor aceptación.

Sin embargo, para acervos muy grandes de datos, la estructura cada vez más utilizada es la basada en pares Llave/Valor (Key/Value) (Jones E., Abadi D., Madden S., 2010): en lugar de almacenar los datos en campos fijos de tablas, se los incluye en los registros con un identificador que señala el significado del dato. Los grandes acervos de datos que se presentan para ofrecer búsquedas en páginas en Internet o en colecciones muy numerosas de datos usan precisamente modelos de ese tipo.

Hace ya 25 años, Bauer elaboró un modelo basado precisamente en este concepto para obtener, de un repositorio de datos, subconjuntos que satisficieran ciertas condiciones, mismos que se pueden llamar criterios de selección o de búsqueda. Este modelo a la larga evolucionó en la base de datos denominada DBB, que, además de estos pares de valores, usa campos fijos y datos sin estructura para almacenar la información. El DBB no es relacional, ni permite su uso con relaciones análogas a las que se usan en bases de datos relacionales. Las consultas se hacen en base a los pares de valores descritos, mediante

operaciones lógicas entre los subconjuntos de registros del repositorio que contienen dichas marcas, como se denominan en DBB.

El DBB ha sido un proyecto de la especialidad de Computación Aplicada hace varios años. De hecho ya ha producido 4 Tesis de Maestría, aunque una de ellas versa sobre un paquete (KBC) que, a pesar de ser imprescindible para el DBB, se maneja por separado a nivel de diseño y programación.

Para determinar si este modelo pudiera ofrecer ventajas en ciertas situaciones, se diseñó un estudio en el que se compararían ciertos aspectos de los datos de una situación de negocios usando una base de datos relacional y la estructura DBB propuesta. En particular, se compararon los tiempos de respuesta de ciertas consultas que pudieran aparecer en una situación de análisis de negocios, y el espacio en disco ocupado en ambas tecnologías. Es importante señalar que - por varias circunstancias - no se trata de un auténtico benchmark. No se incluyeron “todos” los modelos, y no se plantearon distintas situaciones, para hacer la comparación: ésta se limitó a la situación de negocios descrita en este trabajo, y sólo se comparó el DBB con el modelo relacional.

La circunstancia que motivó la definición de este proyecto es que el desarrollo de un paquete como el DBB significa una enorme cantidad de recursos, especialmente humanos. De ese modo, antes de emprender la última etapa pendiente del DBB, que es precisamente la programación, pruebas y documentación completa del producto, se deseaba determinar si un esfuerzo como el que ello implica tendría sentido práctico, puesto que no hay duda que académicamente, lo tiene.

Al proyecto se le agregaron varios objetivos secundarios, donde esto no implica que los resultados obtenidos tendrían menos importancia, sino que el énfasis está en el objetivo principal enunciado más arriba. Entre tales resultados secundarios se pueden mencionar los 4 principales:

- determinar circunstancias adversas para el DBB, es decir, aspectos técnicos o de aplicación que fueran desfavorables comparados con otras bases de datos,
- determinar aspectos del diseño que pudieran ser mejorados;
- encontrar elementos faltantes en el diseño presente del paquete y que deberían ser agregadas;
- y finalmente, lo que se tendría que modificar o agregar al KBC, el paquete en el que se basa parte de la funcionalidad del DBB, como se verá más adelante.

Para cumplir con los objetivos señalados, se definió una situación de análisis de negocios para una librería. Ésta pretende explotar la información de sus clientes y sus ventas para incrementar estas últimas. Para ello, cuando un cliente adquiere un libro, la librería le ofrece otros que – según la información disponible – pudieran ser de su interés. La estrategia seleccionada fue ofrecer al cliente que compra un libro L1

- todos los libros
  - que adquirieron otros clientes que también compraron L1
    - exceptuando los que ya compró el cliente actual.

Se procedió a determinar las actividades que se desarrollarían como parte del proyecto de investigación, y los recursos informáticos (especialmente manejadores de bases de datos y computadoras, pero también dispositivos de almacenamiento) que se necesitarían.

A continuación se conformó el equipo de trabajo, en el que participaba en forma preponderante el autor de esta tesis, pero también otros investigadores, especialmente en los aspectos de diseño del DBB, pero también en lo referente al KBC, que es vital para el funcionamiento del DBB. Esto resultó en otro subproyecto, precisamente definido para producir una versión nueva del KBC, y que dio lugar a otra Tesis de Maestría, la de Nancy Hernández Negrete (Hernández Negrete. Nancy., 2010).

Naturalmente, el director de la investigación también participó en forma activa en varias actividades, especialmente por la carga de trabajo que representaba este proyecto en el diseño de sus diversas componentes, la programación y las pruebas de los programas. Sin embargo, todas las ejecuciones de procesos fueron realizadas exclusivamente por el autor de esta tesis. Lo mismo sucedió con todos los aspectos relacionados con las bases de datos relacionales, mientras que en los programas para la generación de los datos y la implementación en el DBB en ocasiones incluyeron a los restantes miembros del equipo de trabajo.

Se ha dividido el material presentado en esta tesis en capítulos, que reflejan el orden en el que se presentan los diversos tópicos y descripciones. A la introducción de este Capítulo 1, sigue el Capítulo 2 que contiene una descripción de los aspectos relevantes de las base de datos relacionales, pero en cuya introducción se presentan muy brevemente algunas de las tecnologías que están reemplazando a las bases de datos relacionales en la explotación de grandes acervos de datos, especialmente de bodegas (o almacenes) de datos. El Capítulo 3 está dedicado a las características y aspectos principales del diseño del DBB. Una descripción de la metodología utilizada en la investigación, en el Capítulo 4, contiene el detalle de las restantes secciones de la tesis, incluyendo la descripción de la situación de negocios utilizada. El capitulo 5 es sobre la implementación de esta situación de negocios en una base de datos relacional, incluyendo especialmente la descripción de la base de datos misma. El capitulo 6 está dedicado al proceso con el cual se generaron los datos simulados que se usaron en la comparación, y en el siguiente se presentan las consultas realizadas con el modelo relacional, con algunos comentarios sobre la importancia y el impacto de la presencia de índices. En el capítulo 8 se describe la implementación de la misma situación de negocios en las estructuras del DBB: primero se muestran las estructuras de los datos almacenados, y luego se detalla el proceso para la carga de datos a partir de los generados en la BDR. La descripción de las consultas y cómo se formularon y ejecutaron en DBB constituyen el Capítulo 9, mientras que en el siguiente se presentan resúmenes de las comparaciones realizadas, puesto

que fue imposible mostrar todas – fueron muchas decenas de miles de consultas. También se compara el espacio en disco utilizado en ambas tecnologías. Para completar la comparación, en el Capítulo 11 se muestra el impacto que tuvo la duplicación del tamaño del acervo de datos sobre los tiempos de respuesta y el espacio en disco ocupado. Puesto que se contemplaron oras consultas, se incluyeron en el Capítulo 12 varias consultas adicionales, especialmente para ver cómo se formularían en ambas tecnologías. Entre las conclusiones, la principal es que el DBB puede ofrecer una mejor solución para algunas bodegas de datos.

CAPÍTULO 2. LAS BODEGAS DE DATOS Y EL ANÁLISIS DE NEGOCIOS

Introducción

En cada vez mayor medida, las empresas y organismos de todo tipo intentan aprovechar la información de la que disponen para mejorar sus procesos de negocios. Esta situación anteriormente sólo aprovechaba los datos de la propia empresa, pero en los últimos años, a raíz de la conectividad creciente entre grandes acervos de datos, cada vez más se trata de incorporar datos externas, en particular pero no exclusivamente los de otras empresas, agregando éstos a los que contienen los sistemas de información de las empresas.

Al integrar información de diversas fuentes, se presentan varios problemas. Los datos están en estructuras y tecnologías diferentes, tienen nombres o identificadores diferentes para datos que significan lo mismo, y en muchos casos, contienen errores que podrían deformar los análisis que los incluyeran. Esto da lugar al concepto de una Bodega (o Almacén) de Datos, los Datawarehouses (DW), que se puede definir por los atributos deseables, definitivamente encabezados por uno de ellos: que los datos estén “limpios”, que es la terminología aplicada al hecho de que la información sea correcta, completa y sin errores. Además, se pretende que estén preparados para diversas consultas u otros usos, y que proporcionen los resultados en el modo más eficiente posible.

La disponibilidad de estos DW a su vez impulsó dos usos de los datos: la inteligencia de negocios y la minería de datos. Como resultado de la gran evolución de estas dos maneras de utilizar la información, surge otra: el análisis de negocios.

En la siguiente sección, se describen brevemente estos tres conceptos, con énfasis en lo que implican en cuanto al uso de los datos en situaciones de análisis de negocios. Es necesario subrayar que, debido a lo extensivo de esta tesis, se han tenido que resumir otros conceptos y limitar las descripciones de las tecnologías emergentes a un mínimo. De este modo, el lector deberá recurrir a las referencias proporcionadas aquí y, naturalmente, a otras fuentes de información, para obtener detalles y ampliaciones a las descripciones que se incluyeron en este trabajo.

2.1 Los usos típicos de las bodegas de datos

2.1.1 Inteligencia de Negocios

Está enfocada especialmente a la presentación de los datos a los interesados. En especial, la elaboración de informes, cuadros y gráficas que permitan apreciar la situación de un aspecto del negocio, o facilitar ciertas decisiones y la formulación de alternativas posibles. Se puede decir que esta disciplina está enfocada a aprovechar toda la información relacionada con algún fenómeno o circunstancia, de modo que el usuario de la misma obtenga un panorama global de la situación. Sin embargo, hay otra componente fundamental incluida en la inteligencia de negocios, muchas veces subestimada: es la cantidad de recursos necesarios para aprovechar la información. Los informes y las consultas "inteligentes" están enfocados a limitar lo que se presenta a lo indispensable para el uso particular de un informe. Por ejemplo, si se emitirán documentos, limitar el número de los mismos o aún, dividirlos de cierto modo de acuerdo al impacto probable de los mismos.

Un ejemplo de una situación de inteligencia de negocios, que de hecho resultó de otra de análisis de negocios (ver más abajo) fue la de las facturas de una empresa telefónica. Se estudiaron dos aspectos, y sus resultados resultaron en una reducción muy considerable en el costo de impresión de facturas:

- determinar si era necesario imprimir la factura "a color": a muchos clientes les daría lo mismo;
- determinar cuáles materiales de promoción se incluirían con cada una de las facturas: no se ofrecen productos que el cliente ya tiene, y se determina cuáles ofertas serían de interés para el cliente.

Naturalmente, para lograr esto, se hicieron numerosos análisis de los datos de la empresa para determinar los criterios que se aplicarían en el proceso. En el caso particular que describimos (sin mencionar razones sociales) la empresa que ideó y realizó el análisis de negocios consiguió el contrato de impresión de facturas precisamente por esa circunstancia.

2.1.2 Minería de Datos

Esta disciplina se dedica a descubrir relaciones “interesantes” o “útiles” entre los datos. Se pueden dividir los tipos de análisis que usa la Minería de datos en dos grandes grupos:

- formular una relación hipotética y comprobarla con los datos, o determinar que no se presenta tal relación;
- “descubrir” relaciones, aún aquellas que fueran totalmente insospechadas o difíciles de imaginar.

El impacto de la posibilidad de hacer este tipo de estudios sobre la organización de los datos a analizar es que, cuando se diseñan los modelos de datos, no se conocen los “usos” de los datos. Esto resulta en la imposibilidad de determinar los índices a incluir en las bases de datos, o de otros modos de acortar los procesos necesarios para realizar los análisis. Esto también es el caso en el análisis de negocios.

2.1.3 Análisis de Negocios

El uso de los datos tiene muchos elementos comunes con la minería de datos: se formula una relación entre los datos y se comprueba o no. A continuación, se aprovecha esta relación en acciones específicas en donde sean aplicables. La diferencia entre los tipos de análisis que se formulan con los que se hacen en minería de datos es que las consultas o comprobaciones son más planeadas, sin que esto signifique que no surgirán algunas improvisadas o motivadas por necesidades no contempladas inicialmente. Esto se debe a que en general las explotaciones de datos están dirigidas a un fin específico, pero en muchas ocasiones, como resultado de estos análisis surgen otras posibilidades de aprovechamiento de los datos.

Para esto, en lugar de informar lo que sucede en la empresa, ya sea un aspecto de la misma o en forma integral, se trata de aprovechar información de otras fuentes (incluyendo especialmente otras empresas del mismo sector) para proporcionar una visión más amplia o global de una situación.

Es importante señalar que no sólo las empresas usan este tipo de análisis.

Los organismos de todo tipo, y sectores enteros de actividades, cada vez más emplean la información disponible para varios fines, entre los que destacan la eficiencia, la eficacia y el mejoramiento de la calidad de productos y servicios. La medicina ha aprovechado estas técnicas hace años, y la integración de un número cada vez mayor de datos augura un cambio en el modo de atender a los pacientes.

No puede concluir una descripción como la que presentamos sin referirse al tema de la gestión del conocimiento: muchos de los productos que se desarrollan están enfocados a este tema. En este tipo de aplicaciones, los datos son no estructurados, lo que significa que la posición de un valor no indica su significado. Para determinar lo que indica un cierto valor, se puede asociar con sus “vecinos” (como en una oración) o asociarle una etiqueta (tag) que proporciones el significante del dato en cuestión. El DBB se concibió inicialmente precisamente para proporcionar una herramienta fácil y eficiente para asociar el significado a las palabras de textos.

2.1.4 Lo que tienen en común

En un sistema de información, como parte de la definición de sus requisitos, se incluye un estudio profundo de las necesidades de información de los involucrados (actores) del sistema. De este modo, la explotación de los datos es *planificada*: cuando se diseñan las bases de datos y archivos, se contemplan en forma prioritaria estos usos futuros.

En los nuevos usos de los datos descritos, se puede decir que sucede precisamente lo contrario. Como no se pueden prever los usos futuros, no se contemplan en el diseño de las estructuras en las cuales se almacenan los datos. Nos referimos a esta circunstancia como la posibilidad de efectuar “consultas no planificadas” es decir, no contempladas o incluidas en los criterios para el diseño de los modelos de datos.

Estas consultas, por no haber sido contempladas precisamente porque no se conocían, sumadas al volumen creciente de las bodegas de datos, han sido las que han puesto en evidencia a las bases de datos utilizadas para los DW en

cuestión. Cuando una carga de datos demora 2 días, es natural que alguien se pregunte si no habrá un modo de acortar el proceso. En forma análoga, pero que sucede con mayor frecuencia, una consulta puede demorar muchas horas, aunque una duración de muchos minutos puede ser igualmente intolerable cando la consulta es urgente.

Los usuarios han llegado a calificar a sus procesos y sistemas de este tipo como inútiles por esa circunstancia. ¡Si una actualización diaria demora todo un día, los datos nunca estarán disponibles! Y si una consulta específica que se necesita en 5 minutos tarda una hora y media, sucede lo mismo.

Naturalmente, los proveedores de tecnología no podían desaprovechar esta oportunidad. A pesar de que se han formulado muchas tecnologías de bases de datos en los últimos años (de hecho, desde que se concibió el concepto de una base de datos) no había productos que se ofrecían a las empresas. Se describirán algunas tendencias y arquitecturas de datos en lo que resta de este capítulo, no sin antes mencionar que el producto que originó este estudio, el DBB, fue concebido en 1986 precisamente porque Bauer consideraba inadecuadas las estructuras ofrecidas, tanto por la falta de flexibilidad que ofrecían a sus modelos de datos como por el uso de los mismos, que resultaría en tiempos de ejecución exagerados. En algunos casos, los incrementos resultantes del aumento del número de datos no es lineal, sino geométrico. Está de moda en el mundo de los economistas y políticos referirse a tales aumentos como *exponenciales* pero rara vez se da ese caso. Pero si un acervo de datos crece en un factor de 10 y eso resulta en un incremento cuadrático de procesos por un factor de 100, la gravedad es parecida a la que plantearía un incremento exponencial: ninguno de los procesos resultantes sería “factible”.

Se ha dicho, y demostrado, que el poder de cómputo (proceso y almacenamiento) crece en un factor de 1000 cada 10 años. Durante mucho tiempo, esto compensó el incremento de datos procesados en muchas aplicaciones, aunque el poder creciente de computación dio lugar a nuevos usos que – como siempre ocurre – exceden aún la capacidad enorme de proceso de las supercomputadoras. En particular, los datos comenzaron a crecer más rápido que

la capacidad de proceso: hace 10 años se hablaba de Gigabytes con respeto. Hoy los terabytes son pequeños, y pronto lo serán los Petabytes.

Como conclusión de esta sección, reproduciremos nuestra actitud hacia este fenómeno. El poder de “computación” (combinar, calcular, etc.) es adecuado, puesto que con la posibilidad de usar muchas computadoras para realizar un trabajo, y el incremento del número de computadoras que pueden ser utilizadas de este modo, de algún modo u otro se consigue mantenerse al día con los volúmenes. Como se dijo, hay nuevos usos de los datos que rebasan el poder de cómputo en este aspecto, pero no son de naturaleza empresarial o de negocios, sino que se refieren a procesos científicos o aun, de entretenimiento. Lo que ha sufrido un atraso notable, porque el crecimiento no fue el necesario, es la velocidad de transferencia entre unidades de almacenamiento masivo y las computadoras, lo que comúnmente se designa como operaciones de entrada/salida (I/O). Y es precisamente en ese aspecto donde se nota la deficiencia de las computadoras actuales en los procesos de minería y análisis de negocios.

Cabe agregar que constantemente hay esfuerzos y descubrimientos dirigidos precisamente a la velocidad de transferencia (y uso) de datos almacenados en dispositivos. Se decidió no incluir este tema en esta tesis, pero los interesados quizá puedan recurrir a la abundante literatura sobre “in memory databases” (Steve Graves. 2002), en los cuales los datos se transfieren en forma masiva a la memoria de la computadora, donde se usan en el resto de los procesos, pero especialmente a las nuevas tecnologías para almacenar datos

2.2 Características, atributos y usos de Bodegas de Datos

Para los diversos usos de los almacenes de datos, éstos deben presentar ciertas características, que se pueden resumir del modo en el que lo hizo Howard (Howard, 2010) en un trabajo que, si bien estaba orientado a ilustrar las bases columnares, en especial una de ellas, presenta en forma sistemática los atributos que debe tener un DW para prestar un servicio adecuado.

Algunos de los atributos que se consideran para calificar un almacén de datos son los que se incluyeron en esta descripción (la numeración no implica algún orden de importancia), pero naturalmente hay muchos otros.

- El espacio en disco ocupado por los datos. Numerosos esfuerzos para reducir este espacio incluyen el almacenamiento más eficiente usando diversas estructuras, y en especial, la compresión de los datos.
- Los tiempos de respuesta a consultas u otros usos: varias arquitecturas de software, hardware y combinaciones de ambos se han enfocado a proporcionar un uso más eficiente de los datos.
- El costo total de operación y explotación de los datos: incluye adquisiciones, licencias, costos de operación como operadores, consumos de energía y agua, mantenimiento, pero también los costos asociados con el desarrollo y uso de programas informáticos.
- El efecto del incremento del número de datos (escalabilidad): se prefieren las soluciones en las cuales estos incrementos tienen menor impacto sobre el desempeño.
- Los costos asociados a cambios en las aplicaciones o en versiones de la tecnología.
- La posibilidad de agregar usos de los datos sin necesidad de cambiar la tecnología o los proveedores.
- El nivel de conocimientos necesario para usar las bodegas, incluyendo no sólo a los usuarios finales (esto es crítico) sino a los encargados de proporcionar nuevos usos de los datos.
- Las duraciones de procesos de carga u actualización de los datos (ETL.)
- La posibilidad de usar la bodega de datos para transacciones “on line”.
- La permanencia de los proveedores involucrados.

A esta lista Bauer agregó dos atributos, que no son nuevos, pero que muchas veces no son citados como deseables o, en ocasiones, imprescindibles.

El primero es que los programas que se usen para hacer consultas pueden hacer consultas de a pasos (stepwise) es decir, formular una consulta a partir de una anterior.

El segundo atributo deseable que se agrega es que sea posible reducir el número de “registros” usando cierta flexibilidad en los modelos de datos que permita la tecnología empleada. A pesar de que muchos autores dicen que “size matters” (el tamaño sí tiene efecto) lo que constituye una especie de broma por el uso de la misma frase en otros contextos, en general se refieren al espacio en disco ocupado por los datos, índices y otras estructuras empleadas para un DB. Aquí el concepto es diferente: según Bauer, el número total de registros tiene mayor impacto que el tamaño total del acervo. El modelo DBB, por ejemplo, ofrece una gran flexibilidad a la formulación de modelos, que puede ser aprovechada para reducir el número de registros (totales) del acervo.

Esto ha dado lugar a una explosión en la oferta de productos y arquitecturas. Constantemente aparecen nuevos proveedores, o los existentes mejoran sus productos, siempre tratando de eficientar y abaratar los aspectos fundamentales de los DW y de su explotación. Mientras que hace unos años, una versión de un producto podía permanecer varios años, ahora estos tiempos para ofrecer la siguiente versión en muchos casos se miden en meses.

A través del tiempo, se han desarrollado diversos métodos y modelos que permiten almacenar y recuperar información de manera eficiente. Uno de los modelos más utilizados son las bases de datos relacionales postuladas por Codd E.F. (1970) de las cuales se presentan algunos detalles en la siguiente sección. La idea fundamental es el uso de "relaciones". Cada tabla está compuesta por registros (las hileras de una tabla), y campos (las columnas de una tabla). El lenguaje más usado para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje de Consultas Estructurado, el cual es un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Con el paso de los años, las necesidades de información tanto en cantidades como en tiempo respuesta para proporcionar los resultados de una consulta han ido creciendo de tal manera que el modelo relacional se ha hecho ineficiente. Para resolver dicho problema se han desarrollado diversos tipos de bases de datos, entre los que podemos mencionar: transaccionales,

multidimensionales, orientadas a objetos, columnares o verticales, documentales, deductivas, distribuidas, entre otras.

En cada vez mayor medida, las empresas reúnen información de diversos sistemas de información operativos, con datos de otras fuentes (incluyendo las de otras empresas del sector) en un repositorio al que se denomina *data warehouse*, termino acuñado por Bill Inmon (2002), traducido como almacén de datos, o también, bodega de datos. Es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta. La creación de un almacén de datos representa en la mayoría de las ocasiones el primer paso, desde el punto de vista técnico, para implantar una solución completa y fiable de Inteligencia o Análisis de Negocios.

Naturalmente hay muchos modos de almacenar los datos de un almacén de datos. En particular, las diferencias entre ellos consisten en las estructuras que se usan para almacenar la información, entre los cuales, además de las más utilizadas, que son las bases de datos relacionales, se encuentran modelos de tablas en estrella, en copo de nieve, cubos relacionales, etc.

Cada uno de los tipos de bases de datos utiliza sus estructuras específicas para almacenar tanto los datos, como índices que sirven principalmente para acelerar procesos de consulta.

2.3 Las bases de datos relacionales

2.3.1 introducción

Se decidió no incluir una descripción de las bases de datos relacionales en esta tesis. Ello se debió a que, sin conocimientos básicos sobre este tema, un lector no podría apreciar el tema del trabajo ni los detalles e implicaciones del mismo. Baste decir que una base de datos relacional consta de

- una base de datos (nombre y ciertas opciones y parámetros)
- tablas

- los campos de cada tabla
  - los atributos de los campos (nombre, tipo de dato y otros)
- índices definidos para las tablas
- relaciones entre tablas
- restricciones (consistencia y otras)
- procedimientos catalogados, funciones y vistas
- “triggers” asociados a las tablas
- datos para la protección (seguridad) de los datos
- detalles para el mejor desempeño de la base de datos.

Las bases relacionales se ofrecen como gestores de la base (RDBMS, relational database management system) por diversos proveedores o creadores de las mismas. Se pueden dividir, si tal división resultara útil, en las que se ofrecen como productos en venta o arrendados, y los que se ofrecen en forma gratuita (free software.) Adicionalmente, hay productos “open source” lo que significa que sus usuarios tienen la posibilidad de cambiar los programas que componen el paquete. Finalmente, hay productos “open platform”, lo que significa que pueden ser utilizados con cualquier sistema operativo.

2.3.2 Los índices de una base de datos relacional

Se presentan aquí algunas consideraciones relacionadas con los índices de una tabla de una base de datos relacional.

De acuerdo con Rob P. y Coronel C. (2009), un índice es un arreglo ordenado utilizado para acceder lógicamente a las hileras de una tabla. Cada elemento del arreglo será un par (clave, posición) donde la clave es el valor del índice. El valor suele ser de un campo en particular, o una concatenación de campos. Sin embargo, algunas bases de datos permiten calcular índices, donde los valores se obtienen a través de una expresión o función basada en otros valores. La posición es una identificación de la hilera involucrada, que pueden ser el número de fila o una referencia indirecta, por ejemplo, la clave de un índice agrupado (clustered).

A pesar de que esta definición de índice se refiere a las tablas de una base de datos, puede ser aplicada a muchas colecciones de datos de otro tipo, siempre y cuando las claves apunten a un elemento de la colección. Por ejemplo, el índice de un libro apunta a una página o sección que contiene una clave particular.

En cuanto al tipo de índices de acuerdo a su uso, hay índices principales, foráneos (secundarios). Adicionalmente, pueden ser índices únicos (admiten una sola posición para cada valor) o no, es decir, el valor puede ocurrir en más de un registro.

Hay otros “tipos” de índices, aunque se refieren a cómo se usan, no para qué sirven. Hay índices agrupados (clustered), índices “agregados” e índices invertidos, también denominados listas invertidas. Estos conceptos se aclaran más abajo.

Las estructuras de datos convenientes se utilizan para almacenar los valores de tal manera que se mantenga el orden de los elementos durante las actualizaciones. Las consideraciones que deben tenerse en cuenta al elegir la estructura de datos adecuada para un determinado índice son**:** el espacio necesario para almacenar todos sus valores, y el número de operaciones necesarias para actualizar o usar el índice. El tamaño del índice es importante, ya que por lo general se carga a memoria, además del espacio en disco.

Sin embargo, hay otro aspecto fundamental, especialmente cuando el índice se refiere a una colección de datos muy grandes**:** el número de operaciones de entrada-salida (E/S) que se necesitan para usar el índice, ya sea de carga total o parcial en memoria o para otros usos, es crucial, ya que las operaciones de E/S a menudo contribuyen de manera significativa al tiempo total de procesamiento requerido para una tarea en particular. Se utilizan diversas estrategias para reducir el número de operaciones de E/S, pero probablemente la que prevalece es el uso de paginación de algún tipo, donde los datos no se leen de forma individual, sino que se cargan en páginas de memoria. Shapiro J. (2006) define la paginación como el proceso de mover datos dentro y fuera de la memoria física, mientras que Null L. y Lobur J. (2006) lo interpretan como un método utilizado usualmente para la implementación de memoria virtual en donde la memoria principal es dividida en

bloques de tamaño fijo y los programas son divididos en bloques del mismo tamaño (páginas).

Cada vez más se usan combinaciones de varios modos de almacenar los índices. La compresión, para reducir el tamaño de los índices y de ese modo aumentar la cantidad de información que se puede manejar en memoria, se combina con la paginación o segmentación de los datos almacenados. Otra estrategia es usar los GIN (explicados más abajo) que reducen significativamente el espacio en disco para su almacenamiento. Como se verá, el DBB se basa precisamente en este tipo de índices.

De acuerdo con Rob P. y Coronel C. (2009) la mayoría de sistemas de gestión de base de datos implementan índices usando una de las siguientes estructuras de datos:

- **Función Hash:** se utiliza un algoritmo hash para crear un valor a partir del valor de una columna.
- **B-tree (lo llamaremos árbol-B):** es el tipo de estructura más común utilizado en bases de datos y será explicado a detalle más adelante. La descripción se presenta más abajo.
- **Bitmaps:** se utiliza principalmente en aplicaciones con grandes almacenes de datos (data warehouse) o en tablas con un gran número de filas en las que un pequeño número de valores en la columna se repite muchas veces. En Greenwald R., et. al. (2004), un índice implementado mediante un bitmap es aquel en donde cada bit del índice representa un ROWID. Si una hilera contiene un valor en particular, el bit que representa dicha hilera se “prende”, es decir se pone en 1, en el bitmap para ese valor.

2.4 Árboles B (B-trees)

Weiss (1993) define un árbol - recursivamente - como un conjunto de n nodos, el cual puede estar vacío. De tal manera, un árbol consiste en un nodo distinguido R, llamado raíz, y cero o más subárboles, cada uno de ellos unidos por una arista a R. El árbol-B se definió por primera vez en Bayer R. y McCreight E. (1972) aunque hay una referencia precedente de los mismos autores en 1970,

en la cual se establece: Sea $h \geq 0$, un número entero, y k un número natural. Un árbol dirigido T pertenece a la clase $T(k,h)$ de los árboles-B, si T está vacío $(h=0)$ o tiene las siguientes propiedades:

i) Cada camino desde la raíz hasta cualquier hoja tiene la misma longitud h, también llamada altura de T, es decir, $h =$ número de nodos en el camino.

ii) Cada nodo, excepto la raíz y las hojas tiene por lo menos $k+1$ hijos. La raíz es una hoja que tiene al menos dos hijos.

iii) Cada nodo tiene a lo más $2k$ hijos.

Comer D., (1979) no fue el primero en definir un árbol B +, pero lo incluye como una descripción de las variaciones de los árboles-B utilizada especialmente para almacenar índices. En un árbol B +, todas las claves residen en las hojas. Los niveles superiores, los cuales están organizados como un árbol B, sólo consisten en un conjunto de índices (nodos), que permiten localizar las claves de forma más rápida. La Figura 1 muestra la separación lógica de los índices y las claves en un árbol B+. Naturalmente, los nodos y las hojas pueden tener diferentes formatos o incluso diferentes tamaños. Las hojas son por lo general vinculadas entre sí de izquierda a derecha, como se muestra, para permitir un proceso secuencial de las hojas. La lista enlazada de las hojas se conoce como sequence set.

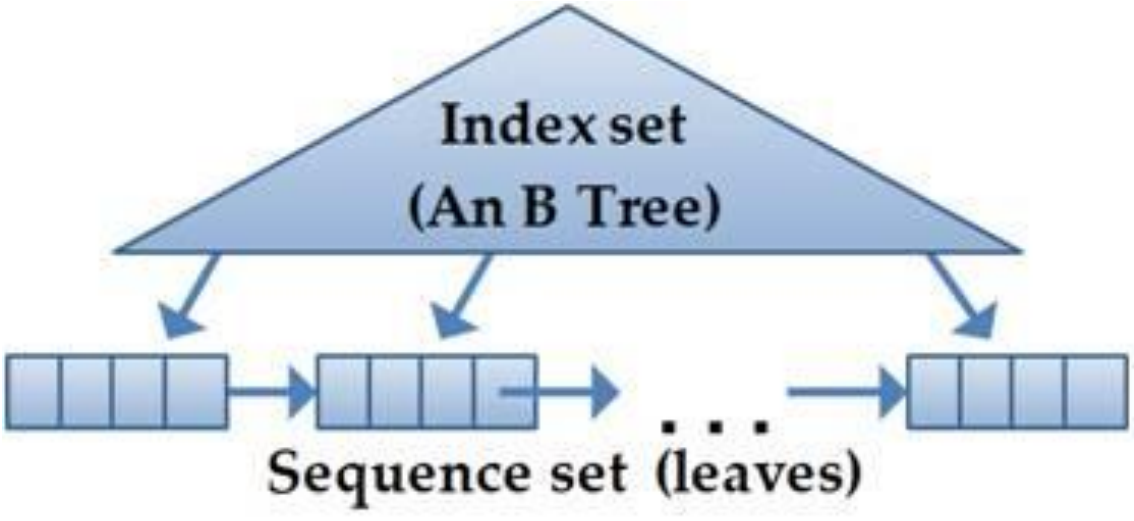


Figura 1. Un árbol B+

Observación: en este trabajo, como en la mayoría de la literatura relevante sobre índices, incluyendo libros de texto, los árboles B+ se denominan árboles B. Esto también se debe a que el KBC no usa árboles B en los cuales no todos las claves están en sus hojas, como es el caso en los B+. Para detalles de estas estructuras y algoritmos para actualizarlas, se recomienda la tesis de N. Hernández citada previamente.

2.5 Índice GIN

La base de datos Adabas (Adabas Comprehensive Data Management System, 2010) fue el primer producto comercial que usó listas invertidas, pero el término de índice GIN no había sido definido, por lo que se presenta la definición establecida en PostgreSQL (The PostgreSQL Global Development Group, 2009): GIN significa *Generalized Inverted Index o* índice invertido generalizado. Se trata de una estructura de índice que almacena un conjunto de pares (*key*, *posting list*), donde un posting list es un conjunto o grupos de hileras en las que ocurre un key. Cada valor del índice puede contener muchas claves, así que el mismo identificador de la hilera puede aparecer en múltiples posting list. Internamente, un índice GIN contiene un índice construido con claves en forma de árbol-B, donde cada clave es un elemento del campo indexado y cada elemento dentro de una hoja tiene la clave y un apuntador que puede ser**:** a un árbol-B (PT, posting tree) o a una lista (PL, posting list), si la lista es suficientemente pequeña.

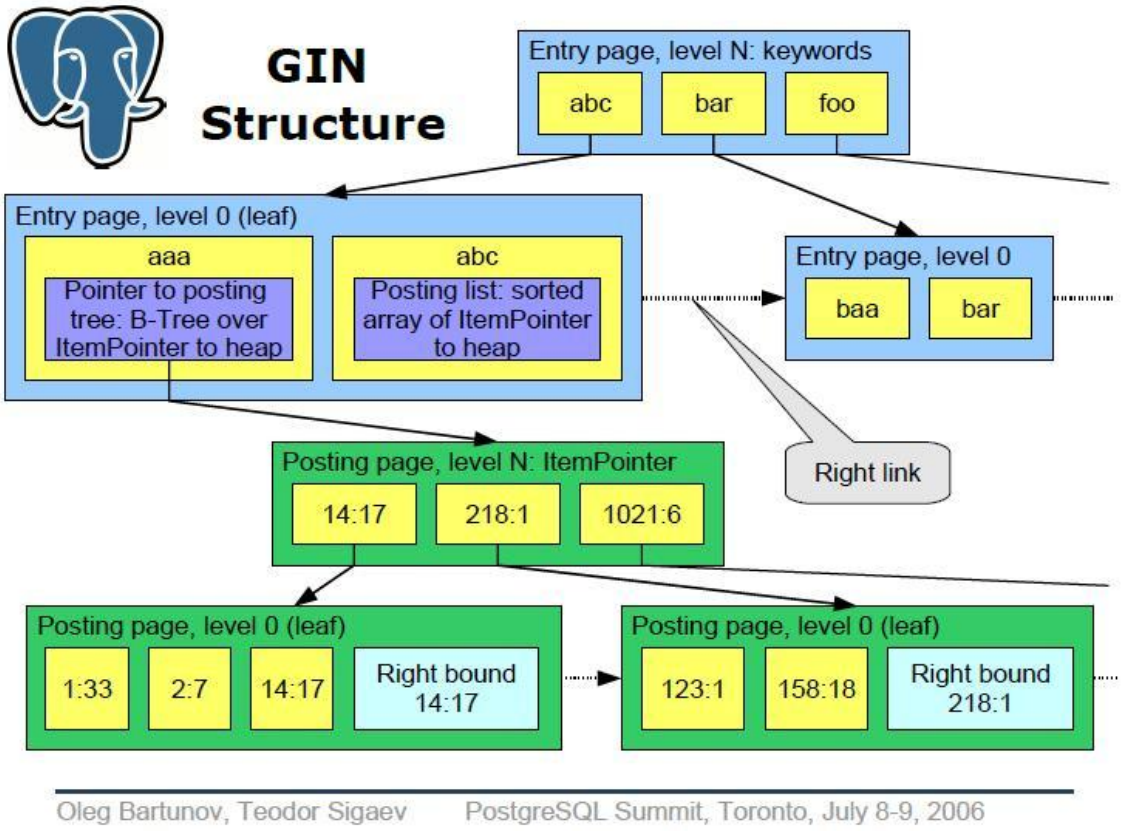


Figura 2. Un índice GIN

En la Figura 2 (Bartunov O. y Sigaev T., 2006.), se muestra un ejemplo de un índice GIN que almacena las claves usando una estructura de árbol-B. En el caso ilustrado, las claves son aaa, abc, baa, bar y foo. En las hojas del árbol están todas las claves, pero además de la clave se almacena un apuntador: en el caso de la clave aaa, es un apuntador a un árbol-B, pero para la clave abc se guarda un apuntador a un arreglo ordenado de todas las hileras que están indizadas con esa clave, donde se usa el término “arreglo” en lugar de lista, que se usó en el término posting list.

El uso de estas estructuras se ha generalizado, ya que son particularmente útiles para los índices en datos no estructurados. Por ejemplo, si en un acervo de datos de diversos tipos, están marcados los objetos que tienen la palabra clave “Rosa”, como se muestra en la Figura 3, al hacer una búsqueda en donde aparezcan todos los elementos que tengan dicha palabra clave, la consulta devolverá elementos que pueden no servir para el propósito de la búsqueda.

Para evitar esta situación, se agrega una etiqueta especifica a las marcas: por ejemplo en el caso del objeto 1, etiqueta=flor y en el objeto 2 etiqueta=apellido, de tal manera que las búsquedas se pueden limitar usando estas etiquetas para aumentar su eficacia. Esto es lo que actualmente se denomina semantic tagging, proceso que se usa principalmente para indizar datos no estructurados, lo que aquí significa que no tienen una estructura específica, como la tiene un elemento de datos cuyo significado o interpretación está dado por su posición dentro del registro, ya sea por el campo o por su ubicación física. Al proveer a un valor de una etiqueta, la interpretación se conoce a pesar de que era desconocida antes de hacerlo. Nos referimos a este tipo de registro como semi-estructurados. Esto quiere decir que se almacenan datos de diferentes tipos y no todos los datos son etiquetados de la misma manera.

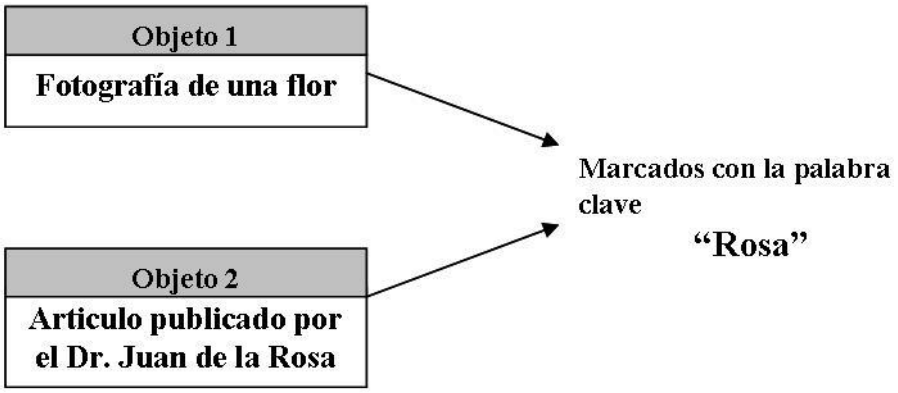


Figura 3. Ejemplo de marcado de palabras

2.6 El uso de bitmaps

Como se indicó, los bitmaps son cada vez comunes en aplicaciones relacionadas con el tema de índices. Diferentes productos pueden utilizar sus propias maneras de almacenar y utilizar un bitmap. En particular, KBC, el producto que usa el DBB para gestionar las marcas, utiliza bitmaps para almacenar el equivalente a una posting list (llamada la lista de instancias de un valor de un contexto).

Los bitmaps están representados por arreglos de enteros (en el KBC, se usan entenros de 4 bytes). Los bits se numeran en cada entero: el bit 0 es el bit menos significativo del entero, es decir, el que representa a 2^0, y el bit 31 es el bit del signo, como se muestra en la Figura 4.

Bit	**31**	**30**	**.**	**.**	**.**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valor del bit	1	0	.	.	.	0	0	1	1	1	0	1	0
+ significativo													**- significativo**

Figura 4. Un bitmap

Cada entero (de 4 bytes) representa 32 números de instancias**:** la instancia está presente en el valor correspondiente a dicho bitmap cuando bit correspondiente está encendido (tiene valor 1). Para un bitmap se especifica el número de instancia representado por el primer bit, ya que éste determina las instancias presentes en el índice como se ve en la siguiente Figura 5, en la que se muestran dos bitmaps, uno correspondiente a las instancias 33 a 64 y el otro a las instancias 97 a 128.

Bit	**31**	**30**	**.**	**.**	**.**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valor del bit	1	0	.	.	.	0	0	1	1	1	0	1	0
Instancias presentes si el bitmap inicia en **33**	44							38	37	36		34	
Instancias presentes si el bitmap inicia en **97**	128							102	101	100		98	

Figura 5. Ejemplo de dos bitmaps

Los algoritmos utilizados para encender o apagar un bit o para determinar si está encendido, usan un arreglo de 32 constantes. Sus valores son los valores de los números enteros en los que sólo el bit indicado por la posición del arreglo está encendido. Llamamos al arreglo Only_bit_set (0-31) y se inicializa la siguiente manera:

Only_bit_set (i) = 2^i para i = 0 hasta 30

Only_bit_set (31) = -2^{31}

Ejemplos del uso de este valor son:

- Si 2235 AND Only_bit_set(17) IS NOT = 0 determina si el valor del bit 17 (dentro del valor 2235) está encendido.

- 2235 OR Only_bit_set(17) encenderá el bit 17 (o permanecerá encendido)

Un arreglo similar de constantes con los complementos, es el que maneja los números en donde todos los bits están encendidos excepto el bit correspondiente al índice dentro del arreglo. Este arreglo se inicializa con las siguientes expresiones:

Only_bit_off (i) = - Only_bit_set (17) – 1 desde I = 0 hasta 30

Only_bit_off (31) = 2^{31} - 1

De esta manera la operación 2235 AND Only_bit_off(17) apagará el bit 17 del número. Una operación usada frecuentemente para determinar si todos los bits de un número están encendidos, es comparándolo NUM=-1**.** Del mismo modo si NUM=0 indica que todos los bits están apagados.

2.7 El lenguaje SQL

El lenguaje SQL es el usado para actualizar y explotar una base de datos. Precisamente esta circunstancia es la que ha hecho populares a las bases de datos relacionales: todas dan servicio a este lenguaje, que se ha estandarizado precisamente para dicho fin. Aunque muchas de los gestores de datos más populares ofrecen las mismas funciones, hay algunas que están más limitadas, de modo que ciertas operaciones que se pueden invocar con el SQL no están disponibles en ellas.

Como el uso del SQL en esta investigación está relacionado con la explotación de los datos, que recibe el nombre general de consultas aunque se pueden usar los datos de otro modo, por ejemplo, para crear otras tablas) diremos que el lenguaje de consultas permite usar datos de varias tablas (de una base o aun de varias de éstas), haciendo uso de relaciones y efectuando las operaciones necesarias para obtener el resultado esperado. En particular, implementa el álgebra relacional vía operaciones entre tablas, o cuando sea posible, índices y estructuras creadas ad hoc para una consulta. El concepto fundamental involucrado es lo que se denomina “JOIN”, que permite combinar datos de dos tablas de varios modos.

2.8 El modelo de datos y los índices

El aspecto más relevante de las bases de datos en el contexto de esta investigación es el desempeño de las consultas formuladas contra el modelo de datos. Sin que ello sea totalmente válido (hay otras causas que afectan desempeño), se puede afirmar que hay dos circunstancias que inciden especialmente en dicho desempeño:

- el diseño del modelo de datos, es decir, cómo se agrupan y relacionan los datos en las diversas tablas, y
- la presencia de "índices" que puedan resultar en consultas más veloces.

Nos basaremos en el siguiente concepto: cuando faltan índices, se degrada el desempeño. Esto sucede porque las consultas o usos no están planificados. Para entender el impacto de la presencia o ausencia de un índice que le podría "servir" a una consulta, es fundamental comprender el concepto de plan de ejecución, que se explica a continuación.

2.9 Plan de ejecución

Cuando se envía una consulta (típicamente con un comando SQL) a un sistema gestor de base de datos, SQL Server en el caso de esta investigación, el servidor debe ejecutar una serie de procesos que permiten recuperar o almacenar información en el menor tiempo posible, manteniendo la integridad de la información. Para lograr este propósito, para cada consulta que recibe, el gestor de la base de datos ejecuta una serie de acciones, que se pueden diferenciar en dos fases:

- La primera consiste en realizar un análisis de la sentencia enviada, para determinar si lógicamente está bien formulada la petición.
- La segunda fase se refiere a aspectos de almacenamiento, es decir se analiza la forma de recuperar la información.

El resultado de los procesos anteriores se llama Plan de Ejecución. A continuación se explican algunas operaciones básicas para entender el plan de Ejecución de SQL Server 2005 (**Gran F., 2008)**.

Table Scan: Significa que el motor tiene que leer toda la tabla. Esto sólo sucede cuando la tabla no tiene un índice "clustered". En algunos casos, cuando es una tabla con *pocos* registros, un Table Scan es la mejor opción, ya que produce poco overhead, es decir, no tarda más leer toda la tabla que procesar un índice. De hecho la tabla puede tener índices y sin embargo el SQL elige usar un *table scan* porque es más eficiente. Pero para tablas con muchos registros, se debe evitar – en lo posible – un Table Scan, ya que es muy costoso.

Para solucionar este problema, hay que ver si la tabla tiene índices y si se están usando correctamente. Lo importante es prestarle atención cuando vemos un Table Scan. Precisamente ésa es una de las operaciones que hace el gestor de la base de datos cuando debe procesar una consulta.

Clustered Index Scan: Esta operación es muy similar a un table scan. El motor recorre toda la tabla. La diferencia entre uno y otro, es que el Clustered Index Scan se realiza en una tabla que tiene un índice Clustered, y por lo tanto, los datos están ordenados en el orden en el que aparecen en el índice, y el Table Scan en una tabla que no tiene este tipo de índice.

Clustered Index Seek: Significa que el motor está usando efectivamente el índice Clustered de la tabla.

Index Seek: Significa que el motor está usando efectivamente el índice Non Clustered de la tabla.

Index Scan: Esta operación se ejecuta cuando se lee el índice completo de una tabla. Es preferible a un Table Scan, ya que el índice es "menor" que una tabla. Esta operación puede ser síntoma de un mal uso del índice, aunque también puede ser que el motor haya determinado que ésta es la mejor operación. Es muy común un Index Scan en un join o en un ORDER BY o GROUP BY.

Bookmark Lookup: Esta es una operación muy importante, El Bookmark Lookup indica que SQL Server necesita ejecutar un salto del puntero desde la página de índice a la página de datos de la tabla para recuperar los datos. Esto sucede siempre que hay un índice Non Clustered. Para evitar esta operación, hay que limitar los campos que queremos incluir en la consulta. Sin embargo, dado que dentro de la estructura interna de un índice non clustered se almacena un puntero

al índice clustered, el bookmark lookup internamente se traduce como un salto a la lectura del índice clustered de la tabla.

Índices agregados: Una de las características nuevas más interesantes de SQLServer 2005, es la posibilidad de usar que incorporan información en la pagina final del índice, campos de la tabla que son externos al índice (el campo no es parte de la estructura del índice): Estos campos no los utiliza el motor a la hora de filtrar y buscar, pero sin embargo, su contenido será copiado a la estructura de la última página del índice. La ventaja de estos índices estriba en que le ahorran al motor del SQL la necesidad de hacer un bookmark lookup, operación bastante costosa. La desventaja es que, al hacer más grande el índice, caben menos registros en una página, lo cual podría llevar a que se tengan que hacer mas operaciones de I/O. Por lo tanto, es necesario hacer una evaluación de costo/beneficio, antes de incluir campos adicionales al índice.

Comentario que no se entenderá en este punto, pero se aclarará tras la lectura de los restantes capítulos: el usar un índice agregado en la tabla VENTAS descrita en la aplicación de negocios documentada en este trabajo, hubiera cambiado muchos tiempos de respuesta en las consultas formuladas en SQL. Al índice Libro-cliente se le hubiera “agregado” la categoría del cliente. Repetimos que esto se aclarará cuando el lector lea la implementación del SOB (sales of books) en la base de datos relacional.

Joins: *Observe que usaremos este término, por su gran difusión en la literatura y los usuarios de las bases de datos.* Un join es la relación entre 2 tablas. SQL efectúa tres tipos de joins. Dependiendo de las características de la consulta y de la cantidad de registros, el motor puede decidir uno u otro. Ninguno es peor o mejor: todo depende de las características de la consulta y del volumen de datos.

Nested Loop Join: Suele ser el más frecuente, y también es el algoritmo más simple de todos. Este operador físico es usado por el motor cuando tenemos un join entre 2 tablas y la cantidad de registros es relativamente baja. También aplica a ciertos otros tipos de joins (por ejemplo los cross joins).

Merge Join: Es otro tipo de join que usan los gestores de bases de datos. Generalmente se usa cuando las cantidades de registros a comparar son

relativamente grandes y están ordenadas de acuerdo a las necesidades de la operación. Aun si no están ordenados, el motor puede predecir que será más rápido ordenar la tabla y hacer el merge join que hacer un Nested Loop Join. En muchas situaciones es frecuente ver que una consulta anteriormente usaba Nested Loop Join y en algún momento lo reemplazó por un Merge Join. La razón es que el volumen de datos aumentó y por lo tanto, resulta mejor usar un Merge join.

Hash Join: Mientras que los Loop Joins trabajan bien para conjuntos pequeños de datos y los merge join para conjuntos moderados de datos, el hash join es especialmente útil en grandes conjuntos de datos, generalmente en datawarehouses. Este operador es mucho más paralelizable y escalable. También se usa generalmente cuando las tablas relacionadas no tienen índice en ninguno de los campos a comparar. Hay que prestar atención si vemos este tipo de operaciones, ya que puede significar un mal uso de los índices. Es importante señalar que los hash joins consumen mucha memoria y SQL Server tiene un límite en la cantidad de operaciones de este tipo que puede efectuar simultáneamente.

Aggregate: Se refiere a agrupar un conjunto grande de datos con un conjunto de datos más chico. Se usan los tipos de aggregate que se detallan a continuación.

Stream Aggregate: Este tipo de operaciones ocurre cuando se llama a una función MIN, COUNT, MAX, SUM, etc. El operador Stream Aggregate requiere que la información esté ordenada por las columnas dentro de sus grupos. El optimizador ordenará los datos que no hayan sido ordenados previamente por un operador Sort anterior. En cierta manera, el Stream Aggregate es similar al Merge Join, en cuanto a las situaciones en los cuales se utiliza.

Hash Match (Aggregate): Hay que tener cuidado cuando vemos este operador. Esta operación también ocurre cuando se llama a funciones del tipo MIN, COUNT, AVG, etc. Así como el Stream Aggregate es comparable al Merge Join, el Hash Match Aggregate es similar al Hash Join. Internamente arma una tabla de hash. En situaciones donde la cantidad de registros es elevada o no están indexadas las columnas por las cuales agrupa la consulta, el motor del SQL elegirá esta operación.

Sort: Como el nombre lo indica (a los que conocen el vocablo), esta operación ordena datos. Solo se hace cuando el campo o los campos que se desean ordenar, no están indexados. A veces esta operación se ejecuta sola, sin que se haya incluido en la consulta el ORDER BY, porque el motor necesita ordenar los datos por alguna razón, por ejemplo, para ejecutar un Merge Join. Pero recordemos que si necesitamos ordenar por un campo indexado y este índice está siendo utilizado, no se ejecuta esta operación.

2.10 Un ejemplo de un plan de ejecución

Para mostrar cómo se arma y ejecuta una consulta en SQL Server, se proporciona un ejemplo de un plan de ejecución de una consulta que se describe a detalle en el Capítulo 7, donde hay numerosos otros ejemplos de este tema

Consulta Básica del SOB en una base de datos con los índices de la tabla VENTAS : Libro-cliente (clustered) y Cliente

La sentencia SQL es la siguiente:

```
SELECT * From libros
WHERE num_lib_dbb
   IN(
      SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente
         IN{
             SELECT num_cliente From clilibro
              Where num_lib_dbb=L1
          }
      EXCEPT
      SELECT num_lib_dbb From clilibro
      Where num_cliente = C1
     )
```

El plan de ejecución es el que se ilustra en la Figura 6.

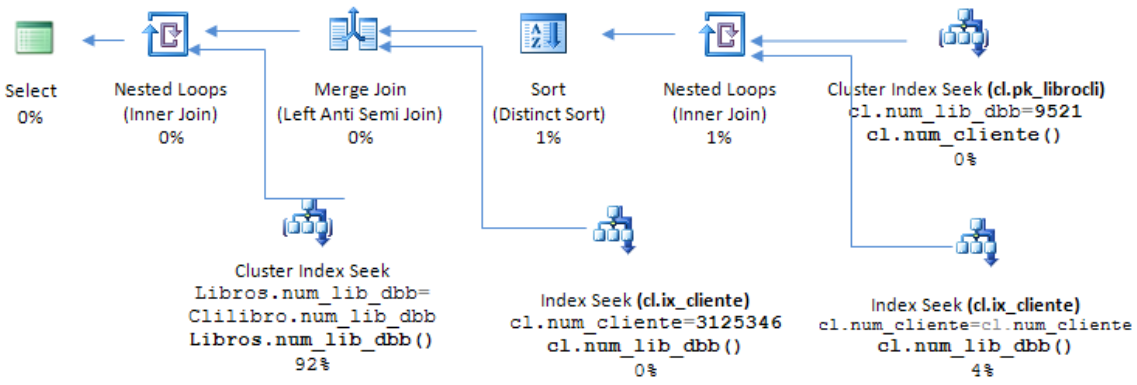


Figura 6. Plan de ejecución de la consulta básica del SOB
con ambos índices, el principal “clustured”

2.11 Bases de datos Columnares

Estas bases a veces reciben el nombre de *verticales.* Lo que las distingue de las bases “horizontales” es que

- en una base de datos horizontal (se puede decir, las tradicionales) los datos de almacenan **por registro** de la tabla (es decir, las filas)
- mientras que en una base de datos columnar se almacenan los datos **por columna**.

Cada vez más proveedores ofrecen este tipo de bases de datos, puesto que ofrecen ventajas para grandes volúmenes de datos. Se puede resumir la principal ventaja y una de las desventajas del siguiente modo:
Ventaja: En lugar de almacenar índices, se arman índices dinámicos para alguna condición impuesta a una columna, mismos que se combinan con los de otras. Los gestores de este tipo crean un bitmap correspondiente a una columna, encendiendo los bits correspondientes al número de hilera en los cuales el valor (o uno de los valores) indicados para el criterio de selección coincide con el de esa posición de la columna. Posteriormente, efectúa operaciones lógicas entre estos bitmaps para obtener el subconjunto de registros resultante.
Desventaja: si una tabla tiene muchos campos, el proceso de actualización puede ser más tardado que en una tabla horizontal. Naturalmente, los proveedores han dedicado mucho esfuerzo a mitigar esta desventaja. No se documentan tales

esfuerzos, pero se pueden encontrar para Vertica, Infosight, Sybase IQ, en las referencias incluidas en este trabajo.

2.12 Bases Orientadas o objetos e hibridas

Las bases orientadas a objetos almacenan objetos (como campos de una tabla) en lugar de sólo permitir valores (un único valor.) Para el caso de los DW, en general tienen poca aceptación puesto que no aportan lo que se necesita para los procesos como los descritos de uso de los DW.

Hay bases de datos denominadas “híbridas”, que a pesar de ser relacionales, permiten el uso de objetos como campos de una tabla. Un ejemplo de este tipo de gestores es POSTGRES (Douglas K., Douglas S.,2003). En particular, la posibilidad de incluir vectores (arreglos) de valores como campos de una tabla resulta en una disminución del número de registros (y tablas) en los modelos. POSTGRES ha mejorado su desempeño y aumentado las funciones que ofrece en forma constante en los últimos años. En particular, lo que resulta de interés es la posibilidad de definir un índice para los vectores incluidos como campos. Esto ya era posible en PROGRESS, pero este manejador nunca ha tenido la aceptación generalizada que obtuvieron las bases de datos más populares.

Como veremos, la posibilidad de incluir vectores e indizarlos (aunque el concepto es un tanto diferente) es parte fundamental del DBB, descrito en el capítulo siguiente y objeto de la investigación que nos ocupa.

2.13 Bases de datos con el concepto de Key-value

Cada vez más se hace imprescindible modificar el esquema con el cual se almacenan datos. El DBB se concibió hace varios años precisamente porque la rigidez impuesta por el uso de campos de una tabla resultaba en bases de datos con muchísimos registros, y precisamente estas cantidades conspiran contra la eficiencia de las bases de datos.

La alternativa es no usar campos que indican precisamente una interpretación o significado, como son los campos de una base de datos

relacional, sino manejar pares de llave-valor: la llave indica el “significado” del dato, y el valor es precisamente eso. En DBB, como se verá, se usan pares de “contexto-valor” para incluir datos, en lugar de ponerlos en columnas fijas.

Esto tiene varias ventajas: permite incluir o excluir estos pares según aparezcan, y además se pueden incluir, en un mismo registro, varios valores de un mismo atributo. Por ejemplo, se pueden asociar a una persona sus hijos como pares “hijo, PABLO”, “hijo, RAUL”, hijo “MARIA”. En una base relacional, esto no se puede excepto si se planean N campos de “hijos”. De hecho, lo normal es crear una tabla de hijos relacionada con la de los padres. Observe que esto hará que, en lugar de un registro (el de la persona) ahora habría 4 (el de la persona en una tabla, y el de sus 3 hijos en la otra.)

En DBB se “indizan” estos pares de contexto y valor. En otras tecnologías, se hace lo mismo con índices invertidos generalizados o con otras estructuras, incluyendo árboles B. Se puede afirmar que casi todos los acervos “enormes” de datos cada vez más usan este modo de almacenar datos, especialmente los descriptores de objetos. La famosa BIG TABLE de GOOGLE usa precisamente este tipo de estructuras para almacenar los descriptores de los sites y de las páginas.

Es importante señalar que no todo es color de rosa en este tipo de indización. Si una entrada (en DBB, marca) puede referirse a varios valores de un mismo registro, al cambiar o eliminar uno de ellos no se puede eliminar la entrada en el índice, o habría que verificar si no hay algún otro campo que tiene ese mismo valor en dicho registro. Esto hace que, en ocasiones, los índices son “Bloom Filters” (Burton H. Bloom. 1970) en el índice se indica que un cierto valor “puede estar”, aunque esto en definitiva no indica que esté, sino que estaba.

2. 14 Tecnologías tipo Appliance Technology

Debido a los tamaños enormes de algunas bodegas de datos, algunas consultas demoran más de lo tolerable, donde esta característica está dictada por el uso de los resultados. Por ejemplo, si una consulta se formula para dar respuesta a una pregunta específica, una duración de 20 minutos puede ser

mayor que la factible: puede ser que la respuesta llegue demasiado tarde para ser de utilidad.

La empresa Netezza (Netezza corporation, 2009). propuso – y ofrece – una alternativa para disminuir los tiempos de proceso basada en una arquitectura de hardware totalmente diferente. Se trata de usar muchos “lectores” de datos (computadoras dedicadas a esta actividad, cada una con su propio disco) para de ese modo disminuir el tiempo necesario para la transferencia de los datos a la memoria. Usa una computadora que recibe la consulta, y ésta distribuye a las restantes computadoras un criterio de inclusión. Cada una de las computadoras “lectoras” le devuelve los datos encontrados en su disco, es decir, los que necesita el proceso que dio lugar a la lectura. Diseñaron un esquema interesante para aprovechar al máximo las computadoras, es decir, para que un proceso en particular ocupe el mayor número posible de las computadoras.

Algunos detalles de la oferta de Netezza

Número de computadoras: se vende en forma modular. El rendimiento es lineal, es decir, el doble de computadoras resultará en la mitad del tiempo de ejecución.

Dispersión de los datos: se definen hasta 4 criterios de separación de los datos (en general son campos de una tabla.) El proceso que graba los datos los dispersa lo más posible entre las computadoras disponibles. Adicionalmente, en cada computadora, tiene información sobre los datos que le llegaron, por ejemplo, el menor y mayor valor de cada una de las variables de dispersión. Esto también lo hace con las páginas o sectores en los que divide los datos en cada disco.

Introdujo varios elementos de hardware y software que permiten efectuar las operaciones descritas. Detalles de lo que hace y ofrece Netezza se pueden ver en la referencia antes citada.

Cabe agregar que NETEZZA no usa índices, de modo que una consulta resulta en una lectura de “todos los datos” de la tabla, pero naturalmente, no se transfieren todos físicamente a memoria, sino que se usan los indicadores para determinar los sectores necesarios.

Posteriormente, han surgido numerosas variantes de este tipo de solución, a las que se llama Appliance Technology. Inclusive, algunas logran efectos similares con componentes de software que reemplazan algunas de las de hardware de Netezza.

2.15 Tecnologías que combinan diversas arquitecturas

Cada vez más aparecen soluciones que combinan diversas características de las bases de datos descritas. Este sector de proveedores de datawarehouses y de grandes bases de datos está creciendo significativamente. Una vez más, se pide una disculpa a los lectores de este trabajo por no proporcionar información adicional sobre este tipo de productos.

2.16 El DBB

Puesto que este modelo es el que se usó para la comparación describe en esta tesis, se ha dedicado el Capitulo 3 a su descripción.

CAPÍTULO 3. DESCRIPCIÓN DEL DBB, UN MODELO HÍBRIDO

3.1 Antecedentes del DBB

Es necesario mencionar algunos antecedentes de este producto, puesto que no nació como una “base de datos”. Inicialmente fue concebido para facilitar el uso de grandes volúmenes de materiales heterogéneos. En los 80’s, cuando se concibió y diseñó la primera versión, aunque había comenzado el uso casi generalizado del cómputo que podríamos llamar gráfico, en oposición a los textos que se manejaban anteriormente (incluyendo en el modo de interactuar el humano con la computadora) era necesario para muchos fines hacer búsquedas entre muchos textos para encontrar un subconjunto de ellos, o alguno en especial. El uso de palabras claves siempre figuró entre uno de los artificios usados, pero la aceptación - sorprendente al principio - que tuvieron las bases de datos relacionales hizo que fueran aprovechadas para estos efectos. Sin embargo, ambos métodos presentaban diversos aspectos que limitaban su uso.

En el caso de las bases de datos, las relaciones “muchos a muchos” o aun “uno a muchos” complicaba los modelos y, en muchos casos, resultaba en consultas complejas y de larga duración. Las palabras clave, por otro lado, resultaban en la inclusión de material no deseado en los subconjuntos de búsqueda.

Por ejemplo, buscar todos los textos que tuvieran un descriptor “Verde” resultaría en muchas cosas que no tenían nada que ver con lo que se estaba buscando. Bauer cuenta que uno de los motivos por incluir los tipos de palabra clave (o, como ahora se llaman, contextos) fue una búsqueda que realizó en Internet con una de los buscadores más usados de esa época. Como resultado de una consulta para encontrar alguien que le vendiera un tipo de sandalias elaboradas en Australia, le ofrecieron un programa - de computadora - para dejar de fumar. Algo no estaba funcionando. Hoy en día esto nos parece gracioso y antiguo, puesto que los buscadores que utilizamos son muy sofisticados y de algún modo “adivinan” lo que estamos buscando, aparte de que tienen mucha información sobre los datos a revisar.

Así nacieron los contextos. En lugar de indicar simplemente una palabra clave, se indicaría el significado que tiene en el texto en el que aparece. De ese modo, podemos tener energía *verde*, el Sr. Verde chocó con un auto verde, el niño pintó el pato de color verde, etc.

La primera tesis relacionada con el DBB resultó en un producto de software denominado Marte, elaborado por Margarita Cruz Millán (Cruz Millán M., 2003). Marte almacenaba textos de cualquier longitud y tenía descriptores, tanto algunos fijos (los campos fijos de la ficha de DBB) como variables, precisamente las *marcas* por tipo de marca. Para encontrar un texto o varios de ellos con ciertos atributos, se formulaban consultas. Se elaboró un producto que manejaba las marcas, el actual KBC aunque en aquella ocasión se llamó de otro modo. El modo de marcado era “colorear” una palabra de un texto con el color asignado al tipo de marca aplicable.

El paquete gestor de las marcas se elaboró al vapor, puesto que no era parte del trabajo de investigación, y se utilizaron diversos artificios para implementar en pocas semanas un producto de enorme complejidad. El KBC se basa casi totalmente en árboles B. Puesto que la puesta a punto de los numerosos usos de estos árboles, en conjunción con otras estructuras que se usan para guardar marcas, era muy laboriosa, se utilizaron los índices de una base de datos relacional (en este caso, ACCESS) para aprovechar los árboles que se usan para dichos índices.

El segundo uso del DBB fue un producto denominado RISP, incluido como una componente de los U-books de Bauer (Bauer Mengelberg J., 2007.). El RISP ampliaba algunas funciones del MARTE, especialmente incorporando las consultas y su ejecución como fórmulas que resultaron de otras dos tesis de maestría de la especialidad de Computación Aplicada en el Colegio de Postgraduados. (Ramírez Galeano E., 2006) y (Lugo Espinosa O., 2006)

Simultáneamente, el mundo no estaba detenido, y muchos otros investigadores estaban ideando y creando nuevas estructuras para complementar o reemplazar lo que hacían las ya conocidas. Estos temas se comentaron en el capítulo anterior. Lo que tenían en común todas ellas y el DBB, era que los

volúmenes de datos habían crecido más rápido que la tecnología, especialmente en un aspecto: la velocidad de las operaciones de entrada y salida. Todos sabían que era imprescindible hacer algo para reducir el número de estas operaciones necesarias para un uso específico de los datos, por ejemplo, una consulta, o descubrir alguna otra manera de evitar el impacto de las operaciones de I-O.

Se describieron anteriormente las llamadas soluciones “appliance technologies”, lidereadas por Netezza, que decidieron resolver este problema con fuerza bruta: si había que leer muchos datos, usarían muchos lectores. En cambio, el otro enfoque consiste en reducir el número de operaciones de I-O necesarias, sea como fuere que se logre este propósito.

La formulación de esta situación se puede resumir de este modo. Dado un conjunto de NNN datos, que pueden ser cualquier cosa pero por ahora hablaremos de registros o unidades de información, se trata de obtener un subconjunto que cumpla ciertas condiciones. Para ello, supongamos que hay que leer NN de los datos (un subconjunto.) Cuanto más pequeño sea NN, menos operaciones habrá que hacer, o por lo menos, en muchos casos sucede precisamente eso.

Ése es precisamente el uso de los índices de una base de datos: permite hacer operaciones sobre el índice, lo que produce un subconjunto de los registros que cumplen los criterios indicados. Si los criterios usan “campos” que no están en ese índice o algún otro, en general habrá que leer todos los datos.

El uso de palabras claves por contexto parecía ofrecer una posibilidad de lograr algo similar a los índices, pero mucho más general. Por lo tanto, se amplió el concepto de DBB: se agregarían campos (como a una tabla de una base de datos) a los textos que tenía, y algunos otros elementos que se expondrán más abajo en este mismo capítulo. De este modo, el DBB era una base de datos como cualquier otra, sólo con características adicionales en materia de flexibilidad en el tipo de modelos que implementaba, pero especialmente en muchos aspectos técnicos.

3.2 los componentes del DBB

Es importante recordar que el DBB es un paquete de software. Las aplicaciones del paquete serán instancias del mismo. De ese modo, una aplicación usa una versión del paquete de software (que tiene algunos datos y parámetros en archivos y bases de datos que se incluyen con el paquete) y tendrá los archivos necesarios para esa aplicación, que residirán en un directorio asociado precisamente con la aplicación en cuestión.

La descripción de los componentes se presenta en secciones numeradas, correspondientes a los siguientes conceptos:

Descripción panorámica del DBB

Los contextos, las marcas y el KBC

Los campos de una UBI

Las clases de UBI

Actualización de datos

Consultas

Cómo se almacenan los datos en DBB

Opciones disponibles para las aplicaciones.

Observe que se ha dejado para el final la exposición de las opciones puesto que se refieren a términos y conceptos que no se hubieran entendido si su presentación hubiera precedido la de estos conceptos.

Siempre que sea posible se mencionarán componentes o conceptos similares de una BDR. La terminología se introducirá a medida que aparezcan las palabras: la primera vez que ello suceda, se las indicará con letra cursiva. Es importante señalar que algunos conceptos y componentes del DBB no se incluyeron en esta descripción puesto que el tema central de la tesis no es el DBB, sino la descripción del proyecto descrito anteriormente. De ese modo, se han omitido algunos aspectos que no intervinieron en la situación de negocios usada para este trabajo.

3.3. Descripción panorámica del DBB

El DBB es un conjunto de elementos de datos que llamamos *UBI* (Unidad Básica de información) o también *ficha* , equivalentes a un registro (o hilera) de una BDR. Las diferencias son muchas, pero el concepto es análogo. Cada UBI tiene elementos de datos, que son de varios tipos, como se detalla en la sección dedicada a dicho tema. La variedad de tipos de campos es mayor que la ofrecida en otras tecnologías, aunque ya han surgido varias que incorporan casi todos los tipos incluidos en el DBB, como se señaló por ejemplo con POSTGRES. Esta variedad de campos ha dado lugar a que se pueda decir que DBB es una base de datos híbrida en el sentido definido más arriba.

Un concepto central de esta tecnología es que se almacenan juntas todas las UBI de una base, y no por “tabla” como sucede en una base de datos. Para ello, cada UBI tiene un número único de identificación. Como veremos, una UBI pertenece a una clase, que es el equivalente aproximado a una tabla de una base de datos. Por lo tanto, una UBI también se denomina una *instancia* de una clase, adoptando la terminología de la Orientación a Objetos.

El otro concepto fundamental es que se definen contextos (que tienen cierta similitud con índices, pero son mucho más generales que éstos) y se pueden *marcar* por *contexto y valor* elementos de datos de una UBI. El DBB se apoya en el paquete de software KBC para administrar estas marcas, que las almacena y proporciona sus usos. Como se verá, el KBC construye un índice invertido generalizado para cada uno de los contextos que se definen. Estas marcas son las que se usan en consultas: se formula una petición a KBC de entregar los números de todas las UBI que contengan una marca en especial, cosa que el paquete hace creando una *lista de resultados*. Estas listas se combinan a su vez con operaciones lógicas, para obtener de ese modo subconjuntos de UBI que satisfacen los criterios indicados.

Como se explicó en el capítulo anterior, un índice invertido generalizado consta de un árbol de valores de un campo (o una combinación de éstos). Para cada valor, se construye una lista de los números de registro (u otro identificador) de la fila correspondiente. Estas listas se pueden almacenar como árboles, como

arreglos o como bitmaps. En el KBC, los valores corresponden a los de un contexto y no a valores de campos. Esto hace que una misma instancia pueda estar en la lista de más de un valor de un mismo contexto.

3.4 Los contextos, las marcas y el KBC

Puesto que las marcas son el concepto central del DBB, se las describe antes que el resto de sus componentes. Aunque tienen ciertas similitudes con los índices invertidos generalizados (GIN) difieren de éstos en un aspecto fundamental: en DBB, cualquier campo (es decir, su valor) puede ser marcado por contexto, de modo que diferentes campos de una misma UBI pueden usar el mismo contexto. Además, los contextos pueden ser utilizados para fichas de diversas clases. De este modo, las marcas pueden ser consideradas generalizaciones de campos indizados.

3.4.1 Los contextos

La definición de un contexto consta de la asignación de un número de contexto y un “nombre”. A éstos se agregan una etiqueta corta y una descripción que pudiera ser necesaria si hubiera contextos con significados parecidos en una aplicación. Además, se indica el valor de un atributo fundamental del contexto: el tipo de valores que admitirá, según la siguiente tabla de códigos de estos atributos que se muestra en la Tabla 1.

Tabla 1. Atributos de un contexto

Código	Significado	Explicación
1	Normal	No hay restricción de valores
2	Único	Sólo puede haber 1 instancia de cada valor
3	De catálogo	No admite marcas de valores para los que no hay una instancia previa
4	De catálogo pero admite “sin”	Igual que el 3 pero puede haber un valor sin instancias
5	Solo valor “1”	Para campos 0/1, sólo se marcan los “1”

Los contextos de tipo 3 y 4, es decir, los llamados “de catálogo” se usan para evitar que se marquen valores que no estén registrados o uniformados. Por

ejemplo, si hay un valor AZUL del contexto “COLOR” no queremos que alguien marque un valor ASUL.

La diferencia entre el atributo 3 y 4 es que, en el caso de este último, se pueden introducir los valores antes de que aparezca una instancia de dicho valor: esto equivale a formular un catálogo de valores admisibles. Para el atributo 3, si alguien desea introducir un valor “inválido”, el KBC informa que no lo acepta y la aplicación decide si el nuevo valor es admisible o no. Esta acción se puede restringir por autorizaciones en el sistema de AC (access control) del DBB.

3.4.2 Las marcas

Una marca (o palabra clave por contexto) en DBB está indicada por tres elementos: **el contexto**, **el valor** y **la instancia**, donde esta última es el número de la UBI. Para almacenar y gestionar estas marcas, DBB usa un paquete de software denominado KBC, desarrollado hace varios años por Bauer pero que nunca había sido publicado hasta el año pasado, en el que se describieron ciertos aspectos técnicos del paquete en una Tesis de Maestría y en un trabajo enviado para su publicación (Hernández y Bauer Mengelberg, 2010).

DBB envía una marca a KBC, la que lo incluye en el GIN correspondiente al contexto. KBC también construye listas de resultados cuando se le solicite proporcionar el subconjunto de instancias que están marcadas con un par (contexto, valor). KBC también efectúa operaciones lógicas entre tales listas de resultados: unión, intersección, diferencia y unión-todos, donde esto último indica que la lista resultante podrá contener repetidos si una instancia está en más de un conjunto de los que se unen.

Para cada contexto, el KBC construye un árbol B con los valores que aparecen. Para cada uno de estos valores almacena (en la estructura conveniente, determinada por el propio paquete) la lista de todas las instancias marcadas con ese par (contexto, valor.) Las estructuras que usa KBC para estas listas son: “L”, un arreglo ordenado de las instancias; “B”, un mapa de bits (bitmap) que es un conjunto de bits que sólo están encendidos cuando la instancia correspondiente a su posición en el bitmap está marcada con ese valor; “I”, un

conjunto de bitmaps intervalo, y finalmente, “T”, un árbol B de las instancias. El KBC usa un árbol diferente para los valores de contextos únicos, donde se incluye el número de la (única) instancia como parte de la hoja.

Ejemplos de marcas

En el ejemplo de Ventas de Libros (SOB) descrito en este trabajo, las palabras claves asociadas a un libro se marcan de este modo: contexto 6 (indicando que se refiere a una palabra clave); el valor: la palabra clave; y la instancia, el número asignado a la UBI que en la cual se describe ese libro. Observe que de este modo, un libro tendrá varias marcas de este contexto (puesto que como se verá, un libro tiene entre 3 y 5 palabras clave). El libro 145, cuyas palabras clave son Arte, Impresionistas, Monet, Biografía, Pinturas, genera 5 marcas: [6, “ARTE”, 145], [6, “IMPRESIONISTAS”, 145], [6, “MONET”, 145], [6, “BIOGRAFIA”, 145] and [6, “PINTURAS”, 145], donde como se nota, se guardan los valores en mayúsculas y se eliminan los acentos.

Estas marcas permitirán recuperar todos los libros que tienen cualquiera de estas palabras clave, si la consulta correspondiente se formulara como la unión de las listas de resultados de cada una de ellas (operación “OR”). Para encontrar el libro 145 – si no se conociera su número de UBI – la consulta se formularía como: encontrar todos los libros que tuvieran las marcas [6, “ARTE”] AND [6, “IMPRESSIONISTAS”] AND [6, “MONET”] AND [6, “BIOGRAFIA”] AND [6, “PINTURAS”]. Si el subconjunto resultante tuviera más de un elemento, es decir, hubiera otros libros que tuvieran exactamente las mismas palabras clave, se necesitaría alguna acción adicional para determinar el libro buscado, por ejemplo, intersecar la lista con otra marca del libro, como podría ser uno de sus autores.

3.5 Los campos de una UBI

El DBB almacena tanto datos estructurados como campos sin estructura. Para datos estructurados, la posición en el “registro” indica su significado o interpretación; para los no estructurados, sucede lo contrario. Cuando se indica el significado de un valor por otro medio, es decir, no por su posición, se lo llamará

semi-estructurado. Por ejemplo, se puede indicar con una etiqueta (*tag*) el significado, o usar otro tipo de artificio. En DBB se puede indicar el contexto de una palabra de un texto marcándolo, lo que se hará coloreando la palabra en cuestión con un color asociado al contexto. Como este modo de marcar no se usó en SOB, no se comentará el marcado de textos. Los datos de una UBI se agrupan del modo que ilustra la Figura 7.

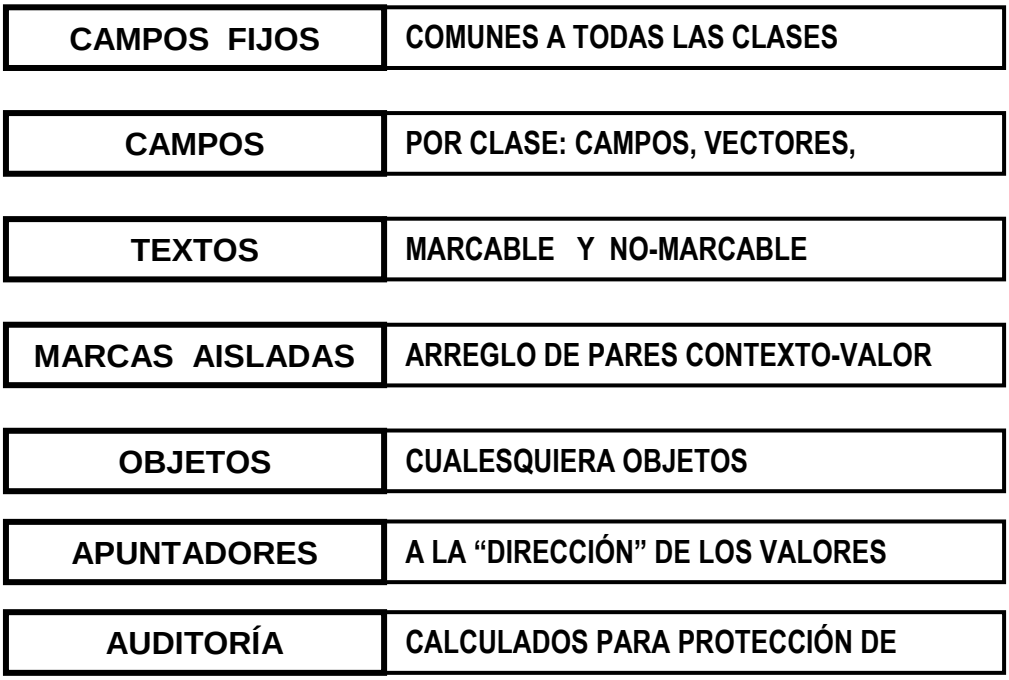


Figura 7. La agrupación de campos de una UBI

3.5.1 Los campos fijos de una UBI

Todas las fichas (como se denominan en forma coloquial a las UBI en la jerga del DBB) tienen los mismos *campos fijos*, que se introdujeron para permitir consultas que involucren fichas de varias clases, y que sirvan para la construcción de "filtros". Estos dos conceptos se explican a detalle más abajo.

Sin embargo, pueden tener significados o usos diferentes de una clase a otra. Por ejemplo, el campo "fecha" (uno de los campos fijos, como se verá) en una clase podría indicar la fecha de actualización de esa UBI, mientras que en otra clase podría ser la fecha de nacimiento de la persona descrita. De otro modo, uno de los campos fijos enteros podría indicar el número de usuario que actualizó la

ficha para todas las clases. Esto permitiría determinar todas las fichas elaboradas por un usuario dado. Es importante señalar que todos estos campos - excepto el número de la UBI y el “titulo” - son optativos, es decir, pueden quedar sin un uso específico para fichas de una clase. En aplicaciones que usan varias clases, también es obligatorio el uso del campo “clase”.

Hay 12 campos fijos, que se muestran en la tabla 2. La numeración se usa para efectos técnicos pero no tiene otro significado. La notación empleada para los tipos de datos son: Ent (x) significa entero de x bytes. Doble = numero decimal de 8 bytes. Str(80) indica una cadena de caracteres de máximo 80 caracteres.

Tabla 2. Los campos fijos de una UBI

#	Campo	Tipo	Descripción o comentario
1	Número de UBI	Ent (4)	El número (único) asignado a la UBI
2	Titulo	STR(80)	Una descripción que aparece en las
3	Clase	Ent (2)	La clase a la cual pertenece la UBI
4	Entero_1byte	Ent (1)	El uso del campo lo determina la clase
5	Entero_2bytes_1	Ent (2)	
6	Entero_2bytes_2	Ent (2)	
7	Entero_4bytes	Ent (4)	
8	Doble	Doble	8 bytes (Decimal)
9	Literal	Str(16)	
10	Fecha	Fecha	8 bytes (fecha y hora)
11	Ano_mes	Ent (2)	Puede ser año (aa, 1 byte) y mes (mm, 1
12	16 V/F	Ent (2)	16 bits con valores 0/1

3.5.2 Campos adicionales

Estos campos corresponden a los campos de una tabla de una BDR. Cada campo tiene un atributo y un nombre, aunque - como se verá en la descripción de clases – también tiene otros atributos. Un campo puede ser

- **Un campo** (número, fecha, si/no, literal, fecha o memo).
- **Un vector de elementos del mismo tipo**: es un arreglo de valores; el vector puede ser de dimensión fija o variable. Por ejemplo, para registrar las ventas mensuales de un año se usaría un vector de dimensión (fija) 12, mientras que para almacenar los productos de un pedido se usaría uno de dimensión variable.

- **Una estructura de vectores “paralelos”**, lo que significa que la interpretación de los elementos de todos ellos es la misma. El ejemplo del SOB usa una estructura para los libros que compró un cliente y el año-mes en el que realizó la compra. Están relacionados por la posición que ocupan en sus respectivos vectores: la fecha de compra del libro que está en posición 7 del vector de “libros comprados” estará en la posición 7 del vector de “año y mes” de la compra. Observe que esto es equivalente a definir un vector de variables de un “type” (en notación Visual Basic.)

A pesar de que no se usó este concepto en el SOB, vale la pena indicar que los campos adicionales se pueden agrupar, de modo que cada uno de los campos pertenece a un grupo. Esto permite limitar ciertas operaciones a los campos de alguno de los grupos. El uso principal de estos grupos es en aspectos de seguridad y confidencialidad, aunque también son muy útiles para clases que tienen muchos campos. Si hay una clase que tiene 300 campos, se pueden ofrecer sólo los de algunos grupos para algún proceso en particular.

3.5.3 Los textos

Se pueden agregar textos a una ficha. Hay un texto marcable (se pueden marcar palabras del texto, en forma manual o automática) y hay un texto no marcable, es decir, que no permite estas operaciones. Ambos textos (o cualquiera de ellos) pueden ser nulos en una dada ficha, si no los necesita. Los textos no tienen restricción de tamaño. Comentario técnico: estos textos se almacenan en formato RTF, pero no en un archivo sino como una cadena en un archivo del DBB.

3.5.4 Las marcas individuales

Se pueden agregar tantos pares contexto-valor como se necesiten a una UBI. El DBB las guarda como un arreglo de pares, y envía las marcas correspondientes al KBC. Todos sus elementos están marcados. Esto permite agregar campos sin estructura, pero con el significado indicado por el contexto. Una vez más, en SOB se ejemplifica el uso de estos campos. Estos pares

(contexto, valor) en otras tecnologías se llaman Key/value. De hecho, es el equivalente a asociar a un valor, vía una etiqueta, un significado o interpretación.

3.5.5 Objetos asociados a la ficha

Se pueden asociar objetos a una UBI. Por ejemplo, la UBI describe un documento y el elemento asociado es un pdf donde está el documento mismo. En otra aplicación, todas las fotografías de un evento se pueden incluir como objetos asociados a su descripción.

El objeto mismo residirá en un archivo de disco, que puede estar en cualquier parte, es decir, no forma parte de los archivos del DBB para la aplicación. Sin embargo, se puede indicar que el DBB lo guarde como propio: se guarda una cadena de bytes idéntica al archivo, pero ahora sí, en un archivo del DBB. Esto es especialmente útil cuando se importan datos de otras fuentes que pueden no estar disponibles en el futuro, pero también se aprovecha para almacenar archivos confidenciales, puesto que el DBB ofrece la deformación de estos archivos para que no puedan ser leídos o vistos sin los programas y autorizaciones correspondientes.

3.5.6 Apuntadores (Punteros) y tamaños

Todos los campos - excepto los fijos - se almacenan por separado. De ese modo, se incluye en la ficha un puntero a la ubicación de los datos, como se explicará brevemente en la sección sobre aspectos técnicos del DBB. También se guarda el número de marcas aisladas que tiene la ficha, y para los vectores de dimensión variable, sus dimensiones para la UBI en cuestión.

3.5.7 Campos de Auditoria

Cada vez que cambie el valor de algún dato de una ficha, se calculan dos números enteros (de 4 bytes cada uno). Antes de usar la ficha en cualquier proceso, se validan estas cifras, para detectar cambios “ilegales”, donde esto significa que se hicieron sin usar los programas del DBB.

El cálculo de estos dos números de protección es secreto, pero se basa en un principio: se arma una cadena de caracteres con ciertos datos de la UBI, y con un algoritmo se calculan los números a partir de los caracteres de la cadena. Cabe agregar que el uso de las cifras de auditoría es optativo: se puede indicar que no se usen para toda la aplicación, o limitar su uso a las fichas de ciertas clases.

3.6 Clases

El siguiente concepto importante del DBB es el de las clases de fichas. La equivalencia con una tabla de una BDR no es total, pero es sumamente útil para entender para qué sirven. En un acervo de datos como los que maneja el DBB habrá datos sobre diferentes cosas: puede ser que se incluyan personas, puestos, libros, bases de datos, ventas, existencias, etc. Como cada uno usa datos diferentes, se crea una clase para cada uno de los tipos de información que se almacena. En el ejemplo del SOB, como veremos, habrá una clase para LIBROS y otra para CLIENTES.

La especificación o definición de una clase es parecida a la que se usa para una tabla de una base de datos, pero incluirá algunas funciones adicionales, puesto que hay que definir “más cosas” que para una tabla, en la cual esencialmente se definen sus campos (con sus atributos) y se pueden agregar índices.

Presentaremos las actividades necesarias para la definición de una clase agrupadas del siguiente modo, con la aclaración de que en general no se harán en ese orden. En el Capitulo 8, se muestran algunas de las formas con las que se define una clase.

1. Especificación de ciertos atributos de la clase misma (número, nombre, etc.)
2. Seleccionar las Opciones que usa esa clase (entre las disponibles): habrá algunas “si o no usa algún tipo de dato” mientras otras se refieren a contextos, numeración de fichas de esa clase, etc.
3. Uso de los campos fijos de una UBI
4. Definición de los grupos de campos (si los usa)
5. Especificar los campos “adicionales” que se usarán

6. Uso de los 16 campos “si o no”
7. Especificación los tipos de objetos asociados (cuando los usa la clase)
8. Marcado automático de campos
9. Indicación de los contextos que usa la clase
10. Opciones de deformación de fichas.

3.6.1 Especificación de ciertos atributos de la clase misma

- Se asigna un número (único, que por ahora no podrá ser mayor a 999) a la clase.
- Se especifica el nombre de la clase (en uno o más idiomas.)

3.6.2 Selección de las opciones que usa esa clase (entre las disponibles)

Hay opciones del tipo “sí o no usa algún tipo de dato” pero también de otros tipos (donde se especifican valores de opciones.) Cuando se seleccionen estas opciones para la clase, ya se habrán indicado las opciones de la instancia: algunas de las opciones de las clases pueden resultar en cambios en opciones, pero sólo en sentido “negativo”. Por ejemplo, para la instancia se indica que podrá haber objetos asociados a fichas, pero para una clase en particular puede ser que no haya tales materiales. El detalle de las opciones disponibles se postergó a la última sección de este capítulo.

Estas opciones harán que, de no usarlas, el programa de actualización no ofrecerá las funciones que permiten usar los datos o conceptos que no se necesitan. Por ejemplo, si en una clase se indica que no se usarán grupos de campos, en los programas no aparecerá ese término.

3.6.3 Uso de los campos fijos de una UBI

En la lista de campos fijos, se indican sus nombres (o se decide no utilizar algún campo para esa clase.) Adicionalmente, se puede indicar una marca automática para cualquiera de los campos.

3.6.4 Indicación de cuáles de los textos se usarán

Una UBI puede contener un texto marcable y otro no marcable. El texto marcable ofrece marcar palabras individuales por contexto, lo que se refleja en que aparecerá la palabra marcada con el color asociado al contexto. Ambos textos no tienen límite de longitud, y se almacenan internamente como cadenas de bytes, pero son una imagen exacta de un archivo con formato RTF que se crea para los textos.

3.6.5 Definición de los grupos de campos (si los usa)

Si la clase usará grupos, conviene definirlos antes de los campos, aunque se pueden agregar grupos posteriormente. Como en SOB no se usaron grupos (no había razón para hacerlo) no se muestran las funciones con las que se efectúan las operaciones, pero se pueden resumir de este modo: en una lista de los grupos ya definidos, se agregan, modifican o quitan grupos.

3.6.6 Especificación de los campos “adicionales” que se usarán

Se agregan (o modifican) campos, análogamente a lo que se hace en cualquier otra base de datos. Para cada campo

- de qué se trata: un campo (un valor único), un vector o una estructura de vectores paralelos;
- se indica un nombre y un tipo de datos, es decir, si es entero, cadena de caracteres, etc. ;
- si se trata de un vector, el sistema pregunta si será de dimensión fija o no;
- se puede indicar una marca automática, especificando el contexto con el que se marcarán los valores del campo o vector;
- si es una estructura de vectores, se indican (en otra forma) los vectores que la conforman.

Cabe señalar que los campos tienen otros atributos, pero no se usaron para el SOB de modo que no se describen aquí.

3.6.7 Uso de los 16 campos “si o no”

Uno de los campos fijos realmente no es “un campo”, sino son 16 campos que se almacenan como un bit cada uno de un entero de 2 bytes. Se pueden usar del modo que sea conveniente por cada clase. Estos campos siempre se marcarán con el contexto apropiado (721 para el primero, 722 para el segundo, etc.) Para usar un campo, se le asigna un nombre (para saber de qué se trata) y se indica, para las fichas de la clase, si el valor de ese campo es verdadero o no. Por ejemplo, se podría usar el campo Si-o-NO (7) para indicar que se actualizó la ficha en esta semana. Cuando ya no sirva ese dato (o cambie la semana) se eliminan los valores de todas las fichas y se indica a KBC que deseche todas las marcas. Observe que sólo se marca el valor “1” (verdadero): no se marcan los ceros.

3.6.8 Especificación de los tipos de objetos asociados (cuando los usa la clase)

En una clase, puede haber fichas que tengan materiales asociados, que son objetos externos a los cuales se refieren los datos de la ficha. En la clase, se indica inicialmente si se podrán asociar tales materiales. En caso afirmativo, hay dos atributos adicionales: será uno solo o puede haber varios materiales para una misma ficha; y si pueden asociarse materiales de diferentes “naturalezas” (textos, videos, imágenes, etc) o todos los materiales serán de un mismo tipo (y en este caso, cuál.) Toda esta información se solicita para no molestar a los encargados de actualizar la información con preguntas o selecciones de opciones superfluas.

Los objetos en general se almacenan en forma independiente. De ese modo, la UBI contiene un apuntador a un arreglo de descriptores de estos materiales asociados a la UBI. En dicho arreglo, se asigna una descripción y algunos otros atributos a cada uno de los objetos, además de su ubicación en disco. En una versión futura del DBB se podrán guardar estos objetos como parte del DBB mismo.

3.6.9 Marcado automático de campos

Ya se indicó en el uso de los campos fijos que se puede indicar que alguno de ellos sea marcado en forma automática con un contexto que se especifica. Lo mismo sucede con los campos asociados, incluyendo los vectores. Si se indica una marca automática para un vector, se marcarán con el mismo contexto todos sus valores.

3.6.10 Indicación de los contextos que usa la clase

Una vez más para facilitar el trabajo de los que usan el DBB, se puede indicar, para cada clase, la lista de los contextos que usa. Esto hace que por ejemplo, al indicar el marcado automático de campos de la clase, no aparezcan todos los contextos definidos para la aplicación, sino sólo los que puede usar la clase. Esta facilidad es optativa.

3.6.11 Opciones de deformación de fichas

El DBB ofrece la deformación del contenido de las UBIs. Se indica (como parte de la definición de cada clase) si los datos de sus fichas serán deformados, y en caso afirmativo, cuáles. Se pueden encriptar los campos adicionales, incluyendo los valores de los vectores, pero no se pueden encriptar los campos fijos. La encriptación se indica a nivel de grupos de campos.

Los algoritmos de encriptación no están diseñados para resistir ataques sofisticados, aunque en general resulta difícil desencriptar información sin el conocimiento de los programas que los deformaron. El énfasis durante el diseño de las rutinas utilizadas fue la minimización del proceso de encriptación y desencriptación.

3.6.12 Cifras de auditoria de las UBI

El DBB siempre (si no se solicita expresamente que no lo haga) calcula una cifra de auditoria en base a los datos de cada UBI. En ocasiones calcula una cifra adicional, pero el concepto es el mismo. Se arma una cadena de caracteres con los valores de los campos, y se calcula un número aplicando cierto algoritmo, con

parámetros determinados *ad hoc* en cada caso. Este número se graba como dato de la ficha. En cualquier uso de la ficha, se determina si la cifra de auditoría es correcta, es decir, se vuelve a calcular y se compara con la que está grabada en la UBI. Si difieren, es porque alguien actualizó algún elemento de la UBI sin los programas correspondientes.

3.7 Actualización de datos en DBB

El DBB se planeó inicialmente para actualización de una ficha a la vez, tanto como resultado de alguna transacción o de otro proceso. La restricción impuesta al diseño del paquete fue que la actualización de las marcas debía suceder en tiempo real, es decir, cada cambio efectuado se reflejaría inmediatamente en consultas posteriores.

Cuando se amplió el diseño para incluir actualización en lotes, se tuvo que cambiar este criterio ligeramente: las marcas no se actualizan individualmente, sino al final de cada lote (o en intervalos intermedios determinados por el proceso en cuestión, especialmente cuando se trata de lotes muy numerosos de UBIs creadas.) De hecho, es lo mismo que sucede en casi cualquier base de datos: los índices no se actualizan en forma individual. Al igual que en otras bases de datos, en DBB puede resultar más eficiente descartar el árbol de valores y crear uno nuevo, cuando hay un número muy elevado de inserciones en un árbol B. Tanto las decisiones como el proceso son responsabilidad del KBC, y no del DBB. Algunos detalles sobre estas actualizaciones están descritos en la tesis de maestría anteriormente citada de N. Hernández.

3.8 Consultas en el DBB

Una consulta basada en un par de valores (contexto, valor) produce una lista de resultados, que es una lista de todos los números de UBI que están marcadas con ese par. Estas listas siempre están ordenadas en forma ascendente. Se pueden almacenar estas listas en disco, para ser usadas en consultas subsecuentes. Esto permite consultas en pasos (*stepwise*), es decir, consultas basadas en resultados de una o varias consultas anteriores. Esto es

indudablemente una de las ventajas de las tecnologías que lo permiten (el DBB es una de ellas) y constituye precisamente una de las desventajas de aquéllas que no lo ofrecen.

Se combinan listas de resultados mediante operaciones booleanas, como se mencionó cuando se describió el paquete en forma general más arriba. Los operandos de una fórmula pueden ser listas formadas anteriormente o pares (contexto, valor): en este último caso, se ejecuta la consulta correspondiente y el resultado se combina con las restantes listas de resultados que aparecen como operandos de la fórmula.

3.9 Filtros de listas de resultados

Tras obtener una lista de resultados, se pueden indicar filtros basados en los campos fijos. Esto hará que se eliminen de la lista las fichas que no cumplan con las condiciones indicadas en el filtro. Esta facilidad se incluye para los casos en los cuales algunos de los campos fijos no están marcados automáticamente. Si estuvieran marcados, se podrían haber incluido operaciones adicionales basadas en los contextos correspondientes. En el DBB, un campo fijo marcado automáticamente no aparece como “candidato” a integrar un filtro de una lista de resultados, precisamente porque se puede lograr el mismo resultado haciendo otras operaciones basadas en el contexto del campo.

Ejemplo: supongamos que en el ejemplo del SOB, que se detalla más adelante, tras obtener una lista de libros como resultado de la consulta, se decide no ofrecer libros que no son de un cierto año, Supongamos también que el año de publicación no se marcó con un contexto. Entonces se puede indicar, cuando aparece la lista de resultados, que se aplique el filtro “año publicación = 2007”.

En la versión definitiva del DBB (que está en desarrollo) habrá modos de incluir otros filtros, basados en valores de campos adicionales no marcados por contexto.

3.10 Otras formas de entregar resultados

Cuando se efectúa una consulta, el KBC devuelve la lista de resultados. En ese momento, el DBB usa esta lista como le convenga al usuario. Lo típico es que aparezca la lista, ampliada con el TITULO de la ficha de cada una de las UBIs de la lista. Se puede solicitar que se incluyan otros datos además del título, pero en esta etapa, sólo usando los campos fijos (para no tener que leer el resto de una ficha).

También se puede solicitar que, en lugar de mostrar toda la lista, se muestren las fichas una tras otra (además de aplicar un filtro como el descrito anteriormente.) Para cada ficha, se muestra "toda la ficha": si tiene muchos campos o marcas, se invocan los datos que no aparecen con acciones específicas. Por ejemplo, si se desean ver los objetos asociados a la ficha, aparece una lista de estos objetos y se pueden "ver" con un click del ratón. Cabe agregar que con este proceso (ver las fichas una por una) se pueden quitar fichas de la lista de resultados (el programa ofrece un botón para eliminar la ficha que se está mostrando.)

En el caso de haber solicitado una base de resultados, cosa que se puede hacer al aparecer la lista o antes) el usuario podrá usar la tabla correspondiente en el modo que desee.

La base de datos de resultados

Se puede indicar que se desea crear una tabla de una BDR con las UBI's de una lista de resultados. Para ello, se indican los campos que tendrá dicha tabla (por default, son todos los campos fijos.) A éstos se pueden agregar campos, por ejemplo correspondientes a ciertas marcas aisladas o a algunos de los campos adicionales. También se puede especificar que se incluya una "columna" del primer material asociado (aparecerá el nombre del archivo correspondiente.)

La versión definitiva del DBB permite elegir un manejador de bases de datos relacionales (entre los que están contemplados en el programa). Se puede usar una base de datos existente o crear una nueva, y lo mismo sucede con la tabla

misma, que puede ser una nueva (se agrega a la base de datos) o usar una existente, en cuyo caso se agregan los nuevos registros a la tabla.

El otro uso de una lista de resultados es combinándola con otras consultas o usos. Se puede invocar un proceso de exportación de datos, enviar ciertos datos para el uso de un paquete de análisis de datos, o un proceso de “cambios masivos” a las fichas de la lista de resultados. Naturalmente, un modo de exportar datos es el descrito en el párrafo anterior: crear o actualizar una tabla de una base de datos.

3.11 Cómo se almacenan los datos en DBB

Las diversas tecnologías de bases de datos usan sus apropias estructuras para almacenar los datos, aunque naturalmente hay algunos elementos comunes en todas ellas. Las bases de datos relacionales guardan los datos por registro. Los campos de cada uno se guardan de acuerdo al esquema diseñado por los respectivos proveedores. El lector interesado puede ver un ejemplo de los esquemas ilustrado para SQL Server en (Celkos, Joe. 2008). En el Capítulo II se mencionaron las bases de datos columnares, que en lugar del agrupar los valores de los campos por registro lo hace por columna, es decir, por campo de la tabla.

El estudio del modo de guardar los datos utilizado por SQL SERVER tuvo dos impactos significativos en el diseño del DBB: por un lado, confirmó que lo que se estaba haciendo tiene similitudes con otros esquemas, y resultó en varias sugerencias que se combinaron con las alternativas formuladas, resultando en algunos cambios a estas últimas.

Para el diseño del modo de almacenar los datos en el DBB, se tomaron en cuenta dos aspectos fundamentales, en este orden:

1. reducir el número de operaciones para actualizar y usar los datos
2. reducir el desperdicio de espacio en disco, donde esto incluye todos los segmentos de los archivos que no contiene información útil.

Como aspectos adicionales tomados en consideración, se pueden mencionar otros dos:

3. La granularidad de los datos, es decir, la posibilidad de no tener que obtener datos que no se necesiten para algún uso específico,
4. Reducir en lo posible el proceso para la preparación de los datos para actualizarlos y para su uso, es decir, su interpretación.

En cuanto al punto número 2, el DBB tiene diversos algoritmos para la reutilización de espacios que quedaron vacantes como consecuencia de una actualización. Esto sucede cuando un segmento de longitud variable se actualiza con una longitud diferente a la que tenía. Naturalmente, el KBC también se diseñó con estos mismos criterios.

DBB almacena los datos por UBI, aunque en varios segmentos. En particular, se guardan los diversos tipos de datos descritos anteriormente por separado. Los principales segmentos son los que se detallan a continuación.

- Se guarda un registro con los campos fijos, los apuntadores y las cifras de auditoría. Este registro tiene la misma longitud para todas las fichas de todas las clases. Los apuntadores indican la dirección física del segmento al que se refieren. En algunos casos serán “ceros”, cuando la ficha no tenga datos de este tipo.
- Para los campos adicionales, se arma un “registro” con todos los de longitud fija, es decir, exceptuando a los vectores de dimensión variable. En este mismo registro se graban los datos necesarios para leer los valores de los vectores de dimensión variable (la dimensión y el apuntador a los valores mismos.)
- Se guardan las marcas aisladas en otra estructura. Lo mismo sucede para los textos, tanto los marcables como los no marcables.
- Finalmente, se guardan los descriptores de los materiales asociados en una estructura independiente de las restantes.

Los diversos segmentos de datos se almacenan en diferentes archivos, cada uno con el diseño adecuado para la naturaleza de los registros mismos. A pesar de que hay un archivo *lógico* único para cada uno de estos tipos, cuando estos archivos crecen se los divide en varios archivos físicos.

No se presentan los detalles de estos esquemas puesto que el DBB no es el objeto de esta investigación. En el futuro cercano se enviará una publicación de

los aspectos técnicos del DBB a una revista especializada, pero no podrá suceder esto antes de que haya una versión depurada y completa de los programas correspondientes.

3.12 Opciones disponibles para las aplicaciones

El DBB tiene tres tipos de opciones, que se ilustran en la Figura 8:

- las que corresponden a la versión instalada del DBB mismo),
- las seleccionadas o aplicables a una instancia en particular
- y las opciones que se ofrecen a las clases.

Las del segundo tipo tienen el efecto de que - en la mayoría de los casos - restringen lo que se ofrece a los usuarios de la aplicación. Si en una instancia se indica que no habrá fichas de diferentes clases (es decir, hay una clase única) el usuario no será importunado con preguntas ni opciones relacionadas con el concepto “clase de UBI”.

Figura 8. Opciones relacionadas con la funcionalidad general (DBB)

El DBB se ofrece en diversas versiones, precisamente de acuerdo a ciertas funciones que ofrece o no a una instalación en particular. Todas las opciones indican las funciones ofrecidas a las instancias del paquete, pero en cada una de ellas se pueden modificar, pero sólo en sentido “negativo”: indicar que no se utilizará una opción disponible. Por ejemplo, puede ser que el paquete instalado permita el uso de contextos virtuales, pero en una instancia no se usarán ese tipo de contextos, de modo que allá se “apaga” la opción. La lista de las opciones presentada aquí en la Tabla 3 no refleja todas las que se implementaron en DBB: algunas se omitieron puesto corresponden a temas que no se describieron de este modelo.

Tabla 3. Las opciones del paquete DBB

1	ofrece uso multi-usuario del DBB (fichas)[no]
2	ofrece selección de rdbms para instancias
3	usa directorio de "paths" (directorios) DEFAULT [SI].
4	Ofrece uso de marcas virtuales [SI]
5	Ofrece cambios masivos[SI]
6	ofrece marcado masivo de fichas existentes [SI]
7	ofrece traducción de diversos nombres de objetos del DBB[NO]

Para cada instancia (aplicación en la que se usa DBB) se seleccionan las opciones que necesita.

Tabla 4. Opciones relacionadas con una instancia

1	usa clases (0= una clase única)
2	Traducción "inmediata" nombres de objetos de clases
3	usa marca 901 (todas las fichas de la instancia)
4	usa marca 902 (las fichas de una clase)
5	usa marcas 701, 702, ETC palabras en orden
6	ofrece listas D (varias listas de valores por registro)
7	ofrece listas I (intervalos de bitmaps)
8	ofrece consultas avanzadas[SI]
9	ofrece consultas catalogadas (si)
10	ofrece crea base resultados a partir de listas [SI]
11	ofrece catálogo de bases datos de resultados
12	ofrece filtros basados en campos adicionales [no]
13	ofrece operaciones muy "caras" [SI]
14	ofrece importación de fichas de otras fuentes
15	ofrece incorporación de datos de otras fuentes a fichas existentes
16	Ofrece usuarios con roles diferentes para diversas clases
17	Usa grupos de contextos

Para cada clase, se pueden modificar las opciones aplicables. El DBB ofrece los valores por omisión de acuerdo a los seleccionados para la instancia.

Tabla 4. Opciones relacionadas con las clases

1	usa billón único (1) o varios billones (0)
2	Usa numeración especial para fichas de la clase
3	ofrece los 2 tipos de texto (1 = si)
4	ofrece marcas aisladas
5	ofrece materiales asociados a fichas
6	Usa grupos de campos adicionales
7	Usa campos adicionales
8	ofrece campos tipo "memo"
9	Usa vectores
10	Usa vectores de dimensión variable
11	Usa estructuras de vectores (vectores paralelos)
12	Usa reglas para nomenclatura de elementos de vectores
13	Usa “idiomas”
14	Usa filtros basados en campos adicionales o vectores
15	ofrece formatos de campos en las clases
16	ofrece opciones confidencialidad a nivel grupos
17	ofrece_marcas_virtuales
18	Nivel de auditoría de campos de la ficha
19	ofrece_multi_valor_una_marca
20	nivel de deformación de datos de fichas
21	ofrece_multi_contexto_un_valor

CAPÍTULO 4. METODOLOGÍA Y HERRAMIENTAS

Introducción: El Objetivo de la investigación

El objetivo formulado para la investigación fue: determinar si, en ciertas situaciones o aplicaciones, el DBB ofrece ventajas sobre otras estructuras usadas para almacenar y usufructuar acervos de datos. Adicionalmente, se trataba de determinar algunos factores relacionados con una aplicación, típicamente una bodega de datos, que aprovecharían estas ventajas.

De ese modo, se definió una situación de análisis de negocios imaginaria, y se implementó usando tanto una base de datos relacional como la tecnología DBB. Se compararían en especial dos aspectos:

- el espacio total en disco necesario para almacenar todos los datos
- los tiempos de respuesta para ciertos tipos de consultas ejecutadas en ambas tecnologías.

Se aprovecharían los procesos para obtener datos de duraciones de procesos tipo ETL para cargar los datos de una fuente externa a las estructuras del DBB. Del mismo modo, se estudiaría el impacto del aumento del volumen de datos almacenados en la bodega de datos en ambas tecnologías.

Se presenta primero la situación imaginada para el estudio, y a continuación se detallan las etapas en las que se dividieron las restantes actividades de la investigación.

4.1 La consulta a ventas de libros utilizada para comparar su implementación en las dos tecnologías

Se concibió una librería imaginaria, que vende libros a sus clientes y registra estas ventas en una base de datos, incluyendo la fecha de adquisición de cada operación. También se almacena información de sus clientes, en especial el país y ciudad, el idioma (principal) en el que adquieren libros, y algunos datos adicionales, que no se incluyeron en el estudio, como el grupo de edad, el género y el nivel de escolaridad. Se denominó a esta aplicación **SOB,** por las siglas en

inglés de Sales of Books. Sin embargo, se amplió el uso de estas siglas a todo lo relacionado con la librería y sus ventas.

Para cada libro que ofrece la librería, además del título, se registran entre 3 y 5 palabras claves (o descriptores), el tema, el idioma en el que fue escrito y otros idiomas en los que venden el libro, el editor y el año de publicación. A esto le se le agrega un resumen o una descripción del contenido del libro.

La finalidad de la situación de negocios es ofrecer a un cliente que adquiere un libro, una lista de otros que pudieran ser de su interés. Se decidió que, para un cliente C1 que adquiere el libro L1, esta lista se elaborara con

- todos los libros que compraron los clientes
  - que también adquirieron el libro B1
    - exceptuando los que ya hubiera comprado el propio cliente C1.

A esta consulta se la denominó la consulta básica, a la que se podrían imponer condiciones adicionales tanto respecto a los clientes involucrados en la consulta o a los libros que se ofrecerían al cliente C1. Aunque los detalles de tales consultas se presentarán cuando se describan a detalle en el Capítulo 11, algunas de ellas pudieran ser:

- sólo incluir libros que tuvieran por lo menos N palabras clave en común con las de B1 (donde N es un parámetro de la consulta);
- sólo incluir libros que estuvieran disponibles en cierto idioma;
- sólo incluir clientes que hubieran comprado por lo menos otro libro del mismo tema que B1;
- otras que se formularían, entre las cuales se incluyó una en especial: no tomar en cuenta clientes que compraron muchos libros (esto se aclarará ampliamente más adelante.)

Se descartaron condiciones sobre el cliente (mismo país, mismo género, etc.) puesto que en DBB no tendrían mayor impacto (es decir, no aumentarían los tiempos de respuesta en más de unos milisegundos) mientras que en la base de datos relacional podrían, en ocasiones, tener un impacto significativo, pero sería similar al de las condiciones ya impuestas en otras consultas.

4.2 Etapas del estudio

Para la realización del resto del estudio, se lo dividió en etapas y se ejecutaron sucesivamente en el orden indicado. De hecho, el resto de esta tesis está organizado precisamente reflejando este orden en el que se efectuaron las actividades. Como se verá, surgieron varios temas de estudio adicionales que se investigaron e incluyeron intercalados entre los que se listan.

- Implementación de la situación en una base de datos relacional
  - Estudio de detalles de los manejadores de bases de datos relacionales contemplados para el estudio: MySql y MS SQL Server.
  - Diseñar el esquema de la base de datos relacional para el SOB.
- Generar datos simulados
  - Diseñar y ejecutar los procesos para generar los datos simulados, incluyendo su almacenamiento en la base de datos
- Las consultas en la base de datos relacional
  - Definir y programar las consultas básicas que se ejecutarían a la base de datos relacional
  - Definir e implementar una base de datos para almacenar los tiempos de respuesta (que serviría también para las consultas en DBB)
  - Ejecutar las consultas
  - Analizar los resultados, y determinar el efecto que tendrían índices adicionales o la eliminación de alguno de los índices de la Tabla de “Ventas” Libro-Cliente
  - Repetir las consultas en las bases resultantes y analizar el impacto de la inclusión de cierto tipo de índices, o el de la ausencia de algún índice
- Implementación de SOB en DBB
  - Diseñar las estructuras en DBB. Preparar los directorios y archivos para el mismo
  - Revisar y probar el paquete KBC, y hacer las modificaciones necesarias si las hubiera
  - Definir, programar y ejecutar los procesos de carga de datos al DBB a partir de los que contenía la base de datos

- Analizar los tiempos de carga de datos (ETL) y determinar si se pueden acortar cambiando los procesos de carga
- Si el resultado del análisis anterior así lo indicara, modificar los programas y repetir la carga de datos
- Si fuera necesario, una vez más se modificaría el paquete KBC

- Las consultas en DBB
  - Definir y programar las consultas en esta estructura
  - Ejecutar las consultas y registrar los tiempos de respuesta en la base de datos mencionada anteriormente
- Comparaciones entre las dos tecnologías
  - Comparar los tiempos de respuesta
  - Comparar el espacio en disco que ocupaban todos los datos en ambas tecnologías
  - Formular conclusiones que se podrían extraer de las comparaciones
- Efecto de la duplicación de datos
  - Repetir todos los pasos de "ejecución de procesos" pero duplicando el número de datos
  - Repetir las consultas y comparar los incrementos debidos a dicha duplicación
- Otras consultas probadas
  - Definición de las consultas que se probarían
  - Detalle de las consultas (sentencias SQL y consultas en DBB)
- Formular las conclusiones del estudio

4.3 Materiales y tecnologías utilizadas

4.3.1 Lenguaje de programación

Casi todos los trabajos se ejecutaron en computadoras con Windows Professional XP, aunque algunas de las computadoras utilizadas operaban bajo VISTA. Sin embargo esto no tuvo impacto alguno, puesto que los programas involucrados operaban indistintamente en ambos ambientes.

Todos los programas necesarios se elaboraron en Visual Basic 6.0. Se contemplaron alternativas (C++ y Punto Net) pero se determinó que no ofrecerían

ninguna ventaja, y el aspecto de depuración favorecía a VB sobre C++, mientras que las posibilidades adicionales que ofrece VB.Net, especialmente en el manejo de clases, no eran necesarias puesto que todos los elementos necesarios se podrían obtener en VB.

Cabe señalar que hubo procesos que se programaron directamente en SQL, especialmente los que operarían cambios al esquema de la base de datos, inclusión de índices, compresiones, pero también algunos de copiado y depuración de datos.

4.3.2 Manejador de bases de datos relacionales (RDBMS)

Se descartó el uso de ORACLE por aspectos de licencia, a pesar de que es una base de datos, al igual que SQL Server, a la cual se le han hecho modificaciones y adiciones sucesivas. Sin embargo, como se decidió realizar todo el estudio en modo mono-usuario, muchos de estos aspectos no tendrían relevancia.

Inicialmente se implementó el SOB en MySQL. Sin embargo, en este último los volúmenes de datos causaban tiempos de ejecución de procesos demasiado largos. Cuando se tropezó por primera vez con un proceso que demoró 2 días y no finalizó, se descartó este producto, que también había mostrado ciertas limitantes en cuanto al número de datos que manejaba. A pesar de que se atribuyó esto a la versión del producto que estaba disponible, se prefirió cambiar de manejador.

Finalmente, se seleccionó el SQL Server de Microsoft. Durante el estudio, por diversas situaciones, especialmente el equipo en el que se ejecutaban los trabajos (inicialmente se utilizó un servidor y luego se ejecutaron los procesos en PC’s, como se describe más abajo) se usaron dos versiones del producto: SQLSERVER ENTERPRISE EDITION 2005 y SQLSERVER DEVELOPER EDITION 2005, seleccionada para todos los procesos.

4.3.3 Dispositivos de almacenamiento

Además de los discos duros instalados en las diversas computadoras utilizadas, debido a que se almacenarían muchas versiones de los datos (especialmente de las bases de datos generadas) se utilizó un disco duro externo de 1 Terabyte. En ciertas etapas del estudio se tuvo necesidad de ejecutar procesos utilizando archivos en esta unidad de discos, pero una vez más, los tiempos de ejecución fueron tales que se tuvo que cambiar de estrategia.

4.3.4 Computadoras utilizadas

Se usaron diferentes computadoras para las diversas actividades del proyecto, pero, con la excepción de un intento por usar un servidor más poderoso (mismo que luego no estuvo accesible debido a circunstancias ajenas al proyecto) todas fueron PC's (algunas de escritorio, otras Laptop.) Si bien no tenían exactamente las mismas capacidades de proceso y memoria, se intentó ejecutar los procesos que darían lugar a comparaciones en los mismos equipos.

Los frecuentes cambios de computadora se debieron esencialmente a tres causas:

- La no disponibilidad de algún equipo en ciertos momentos críticos
- El desempeño no satisfactorio de un proceso en un equipo
- La necesidad de efectuar trabajos en diferentes ubicaciones físicas.

Una circunstancia que afectó negativamente el estudio fue la huelga de 2009, con una duración de 97 días, en la cual fue difícil juntar en un mismo lugar 2 o 3 equipos necesarios para algunas de las actividades. Esto motivó un considerable retraso en el trabajo, lo que causó que la huelga de 2010 volviera a causar problemas similares.

CAPÍTULO 5. IMPLEMENTACIÓN DE LA SITUACIÓN DE NEGOCIOS EN UNA BASE DE DATOS RELACIONAL

Introducción

Los datos generados (de libros, clientes y ventas) se almacenaron en una base de datos relacional. Como ya se mencionó anteriormente se utilizó el manejador SQL SERVER de Microsoft para este propósito.

Se describe en este Capítulo la base de datos utilizada, tanto para la generación de los datos simulados como para realizar las consultas. Para ello, se ha dividido el material en 4 secciones:

1. La base de datos utilizada
2. La generación de datos simulados
3. Los índices incluidos en la tabla Ventas
4. Las consultas formuladas contra la base de datos relacional.

5.1 Las tablas de la base de datos

Se presentan en la figura 9 las tablas de la base de datos, donde se muestran resaltados los campos que componen el índice principal de cada tabla.

Figura 9. Las tablas de la base de datos

Como se verá posteriormente, en el caso de la tabla VENTAS este índice se modificó, y se agregaron otros. El tema de los índices adicionales agregados a la tabla **VENTAS** se comentará por separado en la sección 3 de este Capítulo.

Observación: las tablas adicionales para almacenar datos de libros se introdujeron puesto que un libro puede tener más de un autor, estar a la venta en más de un lenguaje e invariablemente tendrá más de una palabra clave (los libros tienen por lo menos 3 de éstas.)

5.2 Descripción de los campos de las tablas de la base de datos

En lo que sigue se ha utilizado la siguiente notación para los “tipos de datos”:

STR (NN) indica una cadena (texto) de máximo NN caracteres

ENT (M) un entero de M bytes

DBL indica un número flotante de 8 bytes

Fecha Fecha y hora (8 bytes)

Memo es un campo de texto sin limitación de longitud

BIT es un campo que sólo toma el valor 0 o 1

La correspondencia con la notación en SQL SERVER es:

ENT (1) byte

ENT (2) short

ENT (4) integer

ENT (8) long

ENT (8) double

TABLA **LIBROS:** La tabla 5 muestra los campos de la tabla LIBROS de la base de datos. Es importante señalar que el idioma original, en el que fue escrito el libro, se almacena como un campo de la tabla LIBROS, mientras que los idiomas adicionales se guardan en la otra tabla, la de Libro_idioma.

Tabla 5. Campos de la tabla LIBROS de la base de datos

Nombre	Tipo	Nombre	Tipo
id_libro_lib	Str(12)	Cuantos_vendidos	Ent(4)
num_lib_dbb	Ent(4)	Status	Ent(2)
Titulo	Str(50)	Abstract	Memo
Tema	Ent(2)	Num_isbn	Str(10)
subtema	Ent(2)	Num_lib_congress	Str(10)
editorial	Str(24)	Precio	Int
Tipo_libro	Ent(2)	Entrega_en_dias	Ent(2)
Año_publicación	Ent(2)		

TABLA **CLIENTES**

La tabla 6 muestra los campos de la tabla CLIENTES de la base de datos.

Observación: a pesar de que se consideraron componentes de la situación de negocios, características de los clientes tales como género, grupo de edad, nivel de escolaridad se eliminaron del estudio. El motivo fue que su inclusión, aunque permitiría su inclusión en las consultas, no tendría impacto significativo sobre las duraciones de las mismas, en particular, en la relación entre las duraciones en ambas tecnologías comparadas.

Tabla 6. Campos de la tabla CLIENTES de la base de datos

Nombre	Tipo	Nombre	Tipo
Num_cliente	Ent (4)	Numero_tarjeta	STR (50)
Nombre	STR (50)	Banco_tarjeta	STR (50)
Domicilio	STR (50)	Numero_materiales_ comprados	Ent(4)
ciudad	STR (50)	Fecha primer compra	Fecha
Pais	STR (50)	Fecha ultima compra	Fecha
Email 1	STR (50)	Codigo postal	STR (50)
Email 2	STR (50)	Una pregunta	STR (50)
Telefonos	STR (50)	Una rspuesta	STR (50)
Categ_cli	Ent(4)	Su Password	Ent(4)
Tipo_tarjeta	Str(50)		

TABLAS **VENTAS**

La tabla 7 muestra los campos de la tabla VENTAS de la base de datos. En algunas de las bases de datos utilizadas, esta tabla se denomina LIBRO-CLIENTE.

Tabla 7. Campos de la tabla VENTAS de la base de datos

Nombre	**Tipo**
Num_libro	Ent (4)
Num_cliente	Ent (4)
Fecha_venta	Fecha

En la tabla 8 se muestran los campos de las otras tablas de la base de datos, que no necesitan aclaración.

Tabla 8. Tablas adicionales de la base de datos

LIBRO_AUTOR	**LIBRO_PALABRA CLAVE**	**LIBRO_IDIOMA**
NUM_LIBRO	NUM_LIBRO	NUM_LIBRO
AUTOR	PALABRA CLAVE	IDIOMA

CAPÍTULO 6. LA GENERACIÓN DE DATOS SIMULADOS

6.1 Atributos deseados de los datos generados

El objetivo que se formuló para la generación de los datos se puede describir de este modo: se deseaba que las ventas fueran lo suficientemente diversas (en cuanto al número de ventas de los libros y al número de compras de un cliente) para permitir consultas que involucraran diferentes cardinalidades, donde este término se usa en el sentido de la teoría de conjuntos e indica el número de elementos de los conjuntos involucrados en las operaciones (Wikipedia, 2010).

Para ello, se dividieron los libros en 6 tipos de libros, de acuerdo al volumen de ventas, y del mismo modo se definieron 6 categorías de clientes de acuerdo al número de compras que hacen. Se determinaron los parámetros fundamentales de la simulación: el número de libros y de clientes a incluir, los que resultaron en **322,000 libros** y **5,000.000** clientes. El número de ventas a generar no se definió como parámetro, sino que resultaría de los diferentes procesos para la generación de las mismas.

Para almacenar estos y otros datos y parámetros para la generación, se creó una base de datos DATOSGENBASE (se implementó en ACCESS) con las tablas que se detallarán a medida que se las incluya en otras descripciones.

La generación de datos se dividió en tres etapas

- Generación de los libros con sus atributos
- Generación de clientes con sus atributos
- Generación de las ventas simuladas.

6.2 Generación de los libros con sus atributos

6.2.1 Descripción general

Además de los 6 tipos de libro mencionados anteriormente, se definieron 16 temas de libros, 9 idiomas, 80 autores y 400 palabras clave para incluir libros con valores diferentes en cada una de estas variables o atributos. Cada libro genera un registro de la tabla LIBROS descrita en el capítulo anterior, con los siguientes datos, excepto que algunos de ellos se graban en otras de las tablas descritas:

- un tema
- un subtema (no se generó)
- un editor
- el año de publicación
- el editor
- 1,2 ó 3 autores
- un idioma “original”
- de 0 a 2 otros idiomas en los que se ofrece (idiomas adicionales)
- 4 palabras clave
- un título
- un resumen.

Los algoritmos para asignar el valor de cada uno de estos atributos para cada libro generado se describirán más abajo, en el programa (resumido) que se usó para poblar la tabla LIBROS de la base de datos. Se puede adelantar aquí que se usaron distribuciones de probabilidad para cada uno de los atributos, pero en la generación se simularon estas distribuciones asignando sucesivamente valores, de acuerdo precisamente a las distribuciones.

6.2.2 Las tablas creadas para los catálogos

Es importante señalar que muchos de los valores contenidos en estas tablas resultaron de decisiones arbitrarias, es decir, no fueron consecuencia de algún tipo de estudio o investigación.

Para los nombres de los idiomas, autores, editores, temas y palabras clave se usaron cadenas de caracteres constantes seguidos del número en cuestión. Por ejemplo, habría autor 1, autor 2, etc. Observe que de ese modo no se tuvieron que crear catálogos de estos valores.

La Tabla 9 muestra los tipos de libro que se incluyeron en el catálogo respectivo, con los valores que se indican.

Tabla 9. Los tipos de libro y los valores asociados

Campo	Tipo de libro					
	1	**2**	**3**	**4**	**5**	**6**
Mínimo de ejemplares	1	101	501	1001	5001	20001
Máximo de ejemplares	100	500	1000	5000	20000	100000
Cuántos este tipo	100000	150000	50000	15000	5000	2000
Primer número	1	100001	250001	300001	315001	320001
Ultimo número	100000	250000	300000	315000	320000	322000
Ventas mínimas	10	120	501	1001	5001	20001
Ventas máximas	80	300	800	4800	18000	40000
Primer corte	30	200	650	4000	8000	28000
Segundo corte	50	250	750	4500	11000	31000
Incremento	1	1	1	3	10	50

Explicación de algunos de los campos de esta tabla

- El rango de ventas que define al tipo de libro. Está dato por el mínimo y máximo de ejemplares que se venderán de ese libro.
- Primer y último número de libro de este tipo: resultado de la numeración consecutiva de libros de acuerdo al tipo.
- Datos técnicos para el algoritmo que genera el número de ventas para cada uno de los libros

Se decidió no generar números de ejemplares vendidos cubriendo toda la gama de valores posibles. Por ejemplo, para los libros de tipo 1, en lugar de generar libros que se vendieron 1, 2, etc. veces, se decidió limitar las ventas al intervalo mostrado (por ejemplo, en el tipo 1, se generaron libros que se vendieron entre 10 y 80 veces. Los datos “primer corte”, segundo corte” e incremento se usaron para sesgar la distribución hacia los menores números de venta, y se entenderán cuando se explique el algoritmo para el que fueron usados, en la sección donde se describe el programa que lo usa.

La generación se realizó por TIPO DE LIBRO y dentro de cada uno de ellos, por TEMAS, para lo cual se elaboró una distribución del número de libros de cada tipo para cada tema. Los datos se almacenaron en una tabla de la mencionada

base de datos de Datos para generación, con el nombre TIPO_TEMA. La tabla 10 muestra los valores asignados a esta distribución. Observe que no se indican las probabilidades, sino sus valores esperados (y en nuestro caso, reales) para la generación de 322,000 libros. Por ejemplo, la probabilidad de que un libro fuera de tipo 1 y tema 1 era de 1/15. Al aplicar los 322000 libros, resultaron los 20000 que muestra la tabla KK.

Tabla 10. Número de libros por Tipo de libro y Tema

TEMA	**1**	**2**	**3**	**4**	**5**	**6**
1	20000	24500	3000	800	150	200
2	1000	2700	1000	800	150	100
3	3000	4000	2000	800	150	100
4	10000	15000	4000	1000	200	200
5	9000	17000	3000	1500	150	200
6	4000	6000	2000	500	100	100
7	6000	9000	4000	500	100	100
8	8000	12000	7000	800	100	100
9	7000	7000	2000	3500	100	100
10	10000	20000	6000	800	300	200
11	3000	6000	2000	600	300	100
12	1000	1500	1000	1000	400	100
13	8000	11000	6000	600	1000	100
14	3000	3800	1500	500	1000	100
15	6000	10000	5000	500	600	100
16	1000	500	500	800	200	100
TOTAL	**100000**	**150000**	**50000**	**15000**	**5000**	**2000**

Para cada libro generado, se le asignaron entre 1 y 3 autores, usando una distribución de probabilidad

Libros con 1 solo autor 70%

Libros con 2 autores 25%

Libros con 3 autores 5%

Se agregaron 4 palabras clave a cada libro. Inicialmente se había contemplado asignar un número diferente de palabras clave a ciertos libros (entre 3 y 5) pero se decidió que esto no tendría ningún impacto sobre los datos simulados.

Para los idiomas adicionales en los que se ofrece el libro, la distribución fue la siguiente. Se especificaron **3 idiomas "principales"** (idioma1, idioma2, idioma3)

Libros sin idioma adicional	93.8%
Libros con 1 idioma adicional	5%
Libros con 2 idiomas adicionales	1%
Libros con 3 idiomas adicionales	0.2%

Para el idioma adicional, se toman en cuenta solamente idiomas principales, naturalmente diferente al idioma original del libro si este fuera uno de los idiomas principales. Para el tercer idioma en adelante, se genera el lenguaje en forma aleatoria entre todos los idiomas.

De este modo, la generación de los libros se realizó utilizando el programa que se resume a continuación. El programa completo está incluido en el CD anexo a esta tesis. Observe que los comentarios se indicaron en letra cursiva. Las invocaciones de subrutinas o funciones se muestran en negrita. *No se documentan las "conexiones" a la base de datos*

6.2.3 Resumen del programa con el que se generaron los libros

```
For Tipo_de_libro   = 1 to 6
Leer_el_registro_de_la_tabla_tipos_de_libro
menor_num_ese_tipo ' dato del registro
libro_num = menor_num_ese_tipo - 1
numero_de_ventas = dato_de_tabla (Primer corte) - 1

For each Tema   ' de 1 a 16
  Nbooks = Cuantos_libros_ese_tipo_y_tema (tipo, tema)
       ' de la tabla de la base de datos que tiene esta información

  For 1 to Nbooks
    libro_num =   libro_num + 1
    Num_de_ventas = determinar_el_numero_segun_distribucion
            ' esta rutina no se incluyó aquí, pero está en el ANEXO
'   Preparar_el_nuevo_registro
     numero, tipo, tema
     titulo = "LIBRO" & libro_num
     genera_un_resumen   ' cualquiera, puesto que no se usó en la simulación
```

idioma_original *‘ aplica distribución lenguaje principal*
‘ idioma 1 70% idioma 2 20% idioma 3 10%
año-de-publicacion *‘ alternado de 2000 a 2008*
Editor: *alternado de editor 1 a editor 100*
actualiza_tabla_LIBROS_de_la_Base *‘ nuevo registro*
‘ *Genera autor(es) del libro*
Autores_del_libro *“autor + 1” el 81 pasa a 1*
update book_author table (autor**)**

‘ genera autores adicionales
‘ aplica distribution (1,2,3 authors with probabilities 80%, 15%, 5%)
if hay_autores_adicionales then
genera registros adicionales asignando “el siguiente autor” a cada uno de ellos

‘ Genera idiomas adicionales
‘ aplica distribucion numero de idiomas adicionales
(0,1,2,3 additional languges with probabilities 93.8%, 5%. 1%, .2%)
Si procede, generar registros en la tabla Libro-Idioma
‘ *el primer idioma adicional es el siguiente idiomas pricipal (despues del que tiene como original). Si es 4, poner 1.*
Un algoritmo diferente determina los otros idiomas adicionales (serán idiomas no principales).

‘Genera 4 palabras clave
Se generan 4 palabras, cada una con una de las 400 del catàlogo descrito de palabras clave. Para cada palabra clave, se agrega un registro a la tabla Libro_palclave

siguiente tema
siguiente tipo-de-libro

-------FIN DEL PROGRAMA ----------

Comentarios sobre este programa

A pesar de que los procesos fueron prolongados, por la inclusión individual de registros en las tablas, se efectuó el proceso como se describió, pero se hicieron 6 “corridas” (una por cada tipo de libro.) Como se verá, éste no fue el caso para Clientes, como se detalla abajo.

Cuando, posteriormente, se ejecutaron los programas para generar las “Ventas simuladas” se encontraron dos circunstancias: al generar en forma

sucesiva el “número de ejemplares vendidos” de cada libro incrementando el número en 1 hasta llegar al máximo de la categoría, y asignando el menor número cuando el resultado excedía este máximo.

1. El número de ventas generadas era considerablemente mayor al que se había estimado y planeado.
2. Las duraciones de los procesos para la generación de las ventas iba en aumento a medida que aumentaban los números de ventas de los libros, hasta llegar a niveles intolerables (días…) Ello se debía especialmente a la restricción impuesta al número de ejemplares comprados por los clientes que se le asignaban al libro, lo que motivaba una comprobación que alargaba los procesos en forma creciente.

Por lo tanto se tuvo que adoptar un algoritmo diferente. Se basó en los tres datos para cada tipo de libro que se describieron como parte de la tabla Tipo_de_libro.

6.3 Generación de los clientes con sus atributos

6.3.1 Descripción general

Se generaron 5,000.000 de clientes. Para ello, se dividieron en 6 categorías de acuerdo a la tabla de valores que se muestran en la Tabla 11.

Tabla 11. La tabla Categorías de clientes

Categories	**1**	**2**	**3**	**4**	**5**	**6**
Máximo de compras	100	500	1000	5,000	20,000	Más de 20,000
Cuantos clientes	3000,000	1000,000	600,000	300,000	70,000	30,000
Primer # de cliente	1	3000,001	4000,001	4600,001	4900,001	4970,001
Ültimo # de cliente	3000,000	4000,000	4600,000	4900,000	4970,000	5000.000

El cliente tendrá

- un número de cliente asignado en forma consecutiva
- un nombre (“CLIENTE “ & num_cliente

- un PAIS (“PAIS “ & num_pais) El país de cada 10 cliente es el siguiente del anterior, hasta llegar a 10 (el siguiente tendrá país 1)
- código postal
- una ciudad: (10 ciudades por cada país)
- los restantes campos que se iban a incluir fueron descartados porque no se usarían en la simulación (edad, escolaridad y género.)
- email, dirección, tipo y número y banco de la tarjeta de crédito: estos datos, como no se usaron, se generaron sin criterios de distribución puesto que sólo sirvieron para el cálculo del espacio en disco que ocuparían en ambas tecnologías.

6.3.2 El programa para generar los clientes

El proceso de generación de los 5,000,000 de clientes se ejecutó por separado para cada categoría de clientes.

Preparación

Determinación de la categoría para esta ejecución

Conexión a la base de datos

Ejecutar consulta SQL: tabla clientes, selección por categoría de cliente.

Preparación del archivo donde se grabarían todos los “registros” generados.

pais = 0

ciudad_del_anterior = 0

For num_cliente = minimo_esa_categoria to maximo-esa-categoria

Preparar el registro

número de cliente, nombre generado, categoria (de esta ejecución)

CIUDAD = CIUDAD_DEL_ANTERIOR + 1

IF CIUDAD mod 10 = 1 THEN

PAIS = PAIS + 1

IF PAIS = 11 THEN PAIS = 1

END IF

se agregan los restantes campos, incluyendo el nombre (cliente # 1) etc.

Se agrega el registro al archivo (de lote): put al archivo

siguiente cliente

Al final de cada uno de los procesos (para una categoría cada uno) se actualizó la tabla **CLIENTES** de la base de datos con los nuevos registros en modo lote (**BATCH UPDATE**). El índice principal es el número del cliente, y no hay otros índices.

Este paso intermedio (la generación de un archivo con los registros a incluir) fue imprescindible: los procesos para generar los registros individualmente serían muy prolongados (de hecho, realizamos algunos y comprobamos plenamente esta sospecha.) En cambio la actualización de lotes fue muy veloz.

6.4 Generación de las ventas de libros a clientes

6.4.1 Descripción general

Por último, se generaron las ventas de libros a clientes. Este proceso resultó considerablemente más complejo de lo que aparentaba, por los motivos que se exponen a continuación.

Inicialmente se generaban demasiados libros (nos habíamos fijado un número entre 200 y 250 millones de datos como objetivo.) El impacto de un número mayor de libros era que la base de datos crecía, de modo que dificultaba los respaldos y la transferencia de datos de una computadora a otra.

Para cada libro, se tenían que asignar N clientes para la venta en cuestión, donde N era un dato generado durante la creación de los registros de libros. Sin embargo, exigimos a este proceso que respetara “la categoría del cliente”, es decir, que no asignara a un cliente más del máximo de compras de su categoría. Esto exigió agregar un artificio para que se cumpliera esta condición.

Se creó un archivo que tenía un “contador” de los libros comprados por cada uno de los 5,000,000 de clientes. Cada vez que se asignaba a un cliente una compra, se incrementaba ese contador en 1. Antes de asignar un cliente a una venta, se comprobaba que el cliente no excedería el máximo, y si esto sucedía, no se lo tomaba en cuenta.

Este proceso (el de encontrar un cliente que no sobrepasara ya el número máximo de compras correspondiente a su categoría) prolongó tanto las

ejecuciones (con los números de venta generados inicialmente) que tuvimos que modificar estos números en la generación de libros como se mostró antes.

La determinación de los "cortes" e incrementos usados fue una investigación aparte, donde se tomaron valores esperados de ventas generadas y, vía "what if" se determinaron los valores definitivos, que reflejaban las probabilidades asignadas al número de ventas para libros de una categoría. La tabla 12 muestra esta distribución para libros de tipo 1. Observe que las probabilidades mostradas corresponden a cada uno de los valores del rango correspondiente de ventas. Las distribuciones para los otros tipos de libro no se incluyeron aquí. Se usó una hoja de cálculo con las distribuciones.

Tabla 12. Distribución del número de ventas para libros del tipo 1

Valores	**0 a 9**	**10 a 30**	**31 a 50**	**51 a 80**	**81 a 100**
Prob (%)	**0**	**2.25**	**1.5**	**0.75**	**0**

También se indicó la proporción de las ventas por tipos de libro a clientes de diversas categorías. La tabla 13 muestra la distribución (indicada como porcentajes) de ventas de libros por tipo a cada categoría de clientes.

Tabla 13. Porcentajes de ventas de libros de un tipo a clientes de las 6 categorías

Tipo de libro (máximo de ventas)	Porcentaje de ventas de libros de cada tipo a clientes de la categoría indicada					
	1	**2**	**3**	**4**	**5**	**6**
1 – 100	22	38	20	10	5	5
2 - 500	15	20	25	15	10	15
3 – 1000	15	20	20	15	10	20
4 – 5000	12	20	18	15	15	20
5 – 20000	15	25	15	20	15	10
6 – más	10	15	25	25	15	10

Cabe señalar que estos porcentajes no pretenden reflejar alguna situación real o siquiera plausible. Esto no tuvo ningún impacto sobre las comparaciones para las cuales se generaron los datos.

6.4.2 Resumen del programa que generó las ventas

Se presenta a continuación una descripción del programa que generó las ventas de libros a clientes, con todas las reglas y restricciones impuestas. El programa completo se incluyo en modo fuente en el CD Anexo.

Una vez más, se generaron las ventas por tipo de libro, en lugar de una corrida única para todos ellos. Aquí tampoco era conveniente actualizar la tabla Libro_cliente con cada registro generado, de modo que se usó un archivo temporal, mismo que se usó para actualizar la tabla de la base de datos con BATCH UPDATE.

Preparación

Se conecta la base de datos del SOB

Se determina el tipo de libro de esa corrida (se teclea en una forma)

Se crea un RECORDET con todos los libros de ese tipo de libro (ordenado por número de libro) con el comando

```
Select num_libro, numero_de_venas from LIBROS
where libros.tipo_de_libro = tipo-de-la-corrida
order by num_libro
```

Se carga a memoria la tabla Categ-Tipo que tiene los porcentajes descritos anteriormente (sólo se cargan los datos para el tipo de libro de la corrida.)

Se crea (la primera vez) o abre el archivo numero_de_compras_del_cliente tiene un arreglo de 5,000.000 de números enteros.

La primera vez, se inicializa con 0 (ceros).

El archivo se dividió en varios registros, cada uno con 100.000 clientes.

Para cada libro (del Recordset)

NVENTAS = numero-de-ventas (del registro del recordset)

‘ A continuación se *calcula el número de clientes de cada una de las 6 categorías que “compraron” ese libro, aplicando la distribución cargada a memoria de la tabla Categ-Tipo.*
*NV (categ) = porcentaje (categ, tipo) * NVENTAS*
Se redondea el número de ventas hacia arriba.
El número de ventas para la categoria 6 se obtiene por diferencia entre NVENTAS y los numeros determinados para las 5 categorias anteriores.

For categ = 1 to 6

Para indice = 1 to NV (categ)

```
        encontro_cliente = false
        while not encontro_cliente
      num_cli = num_cliente_anterior (categ) + 1
      encontro-cliente =(compras_del_cliente (num_cliente) < maximo-categoria)
      wend  (fin del while de encontro cliente)
‘  ese es el cliente elegido
‘ crea registro ventas    num_libro, num_cli
genera fecha (aumenta en 1 el dia)
contempla fechas de 1 de enero, 2000 al 4 de abril de 2008)
‘ Motivo: ese fue el día de la ejecución de los programas

graba el registro en el archivo para ese lote
actualiza el dato
        compras_del_cliente (num_cli) = compras_del_cliente (num_cli) + 1
next categ
next book
     -----  FIN DEL PROGRAMA -----
```

Al final de cada corrida (para ese tipo de libro) se actualizó en modo BATCH UPDATE la tabla LIBROS-CLIENTES de la base de datos del SOB. Se incluyeron los dos índices de la base 1 (Principal: Libro & cliente, secundario: Cliente)

El número de ventas generadas de este modo (para los 6 tipos) fue de **203,000.000** de ventas.

6.4.3 Comentarios sobre el proceso de generación de datos

Se decidió generar la tabla con los índices descritos. Como se verá en la discusión del impacto de la presencia de ciertos índices sobre las duraciones de ciertas consultas, posteriormente se generaron distintas versiones de la base de datos, precisamente reflejando la presencia de otros índices de esta tabla.

Cuando se ejecutaron los procesos, debido a la duración de éstos, se hicieron modificaciones al programa, algunas de las cuales se describieron anteriormente, y otras solamente mejoraron la eficiencia computacional. Con estos cambios, se logró una disminución muy significativa de los procesos de carga de datos a la tabla de ventas (Libro-Cliente). Estos cambios a su vez requirieron un estudio de varios aspectos de las bases de datos relacionales que usualmente no conocen los usuarios de las mismas, pero también la creación de algunos algoritmos de selección de clientes para un libro.

CAPÍTULO 7. LAS CONSULTAS EN LA BASE DE DATOS RELACIONAL

Introducción

Se describen las consultas que se seleccionaron para efectuar las comparaciones. Se muestran las sentencias en SQL con las que se realizaron las consultas. Por último, se analizan los impactos de la presencia de índices en la tabla principal involucrada (CLILIBRO) y se muestran los planes de ejecución de algunas de las consultas.

7.1 Las consultas

Se decidió denominar “consulta básica” a la que dio lugar a la situación de negocios SOB:

Dado que el cliente **C1** adquiere el libro **L1**

- Determinar la lista de todos los libros - que compraron todos los clientes y que *(también)* compraron el libro **L1 -** excepto los que ya adquirió el cliente **C1**.

A esta consulta se le agregó una condición que se incluyó con el propósito de determinar el impacto que tendría sobre las duraciones (en ambas tecnologías) de las consultas resultantes. De ese modo, se ejecutaron series de consultas con y sin esta condición adicional: Excluir a los clientes de categoría 6.

7,2 Las sentencias SQL

La sentencia SQL sin la condición adicional:

```
SELECT * From libros
WHERE num_lib_dbb
   IN(
      SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente
         IN{
              SELECT num_cliente From clilibro
               Where num_lib_dbb=L1
           }
      EXCEPT
      SELECT num_lib_dbb From clilibro
      Where num_cliente = C1
     )
```

La sentencia SQL con la condición adicional (no tomando en cuenta los clientes de la Categoría 6) Para esta consulta se crearon dos versiones para armar la lista de clientes que compraron un libro.

- La primera de las consultas utiliza la operación de Intersección entre el total de clientes que compraron el libro L1 y la lista de clientes que no pertenecen a la categoría 6 (compran muchos libros).
- La otra obtiene la lista de clientes, quitando de la lista de clientes que compraron el libro L1 aquellos clientes que pertenecen a la categoría 6.

Usando la cláusula INTERSECT de SQL, la consulta es;

```
SELECT * From libros
   WHERE num_lib_dbb IN(
   SELECT DISTINCT num_lib_dbb From clilibro
       WHERE num_cliente IN (
     SELECT num_cliente From clilibro Where num_lib_dbb = L1
       INTERSECT
       SELECT num_cliente From clientes
       Where clientes.categ_cli < 6)
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1)
```

Usando la cláusula EXCEPT también para excluir la categoría 6, la sentencia SQL es:

```
SELECT * From libros
   WHERE num_lib_dbb IN (
   SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente IN (
     SELECT num_cliente From clilibro Where num_lib_dbb = L1
     EXCEPT
     SELECT num_cliente From clientes Where clientes.categ_cli = 6)
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1)
```

Se podría suponer que el uso de EXCEPT resulta más eficiente. Sin embargo el uso de INTERSECT produjo mejores tiempos de respuesta como se muestra en el siguiente gráfico de la Figura 10.

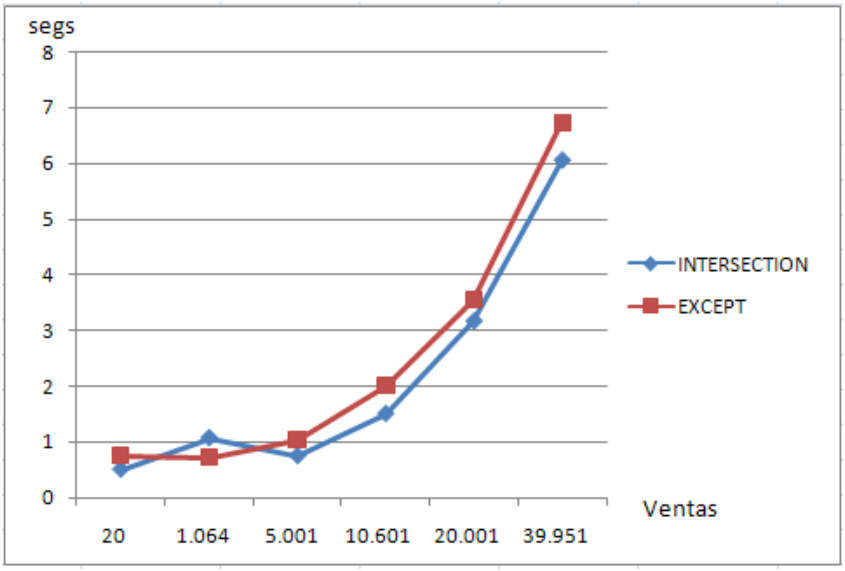


Figura 10. Comparación del uso de EXCEPT e INTERSECT

7.3 Rutinas con las que el programa arma la sentencia para un libro y un cliente

Observación: Se describen las rutinas del programa que ejecuta las consultas en lenguaje Visual Basic.

```
Private Function arma_sentenciaSQL
        (num_libro As Long, num_cliente As Long) As String

Dim PS As String
  PS = "SELECT * " & _
         "From libros " & _
         "WHERE num_lib_dbb IN ( " & _
                        "SELECT DISTINCT num_lib_dbb " & _
                        "From clilibro " & _
                        "WHERE num_cliente IN ( " &
        ARMA_PARTE_SOURCE_DE_LISTA_DE_CLIENTES
        (num_libro, incluye_tipo6) & ")"
        If num_cliente > 0 Then
       PS = PS &  " EXCEPT " & _
                "SELECT num_lib_dbb " & _
                "From clilibro " & _
                "Where num_cliente = " & num_cliente
End If
PS = PS & ")"
arma_para_select = PS
End Function
```

```
Private Function ARMA_PARTE_SOURCE_DE_LISTA_DE_CLIENTES(num_libro As
Long, incluye_tipo6 As Boolean) As String
Dim PSPARTE As String
PSPARTE = "SELECT num_cliente " & _
                "From clilibro " & _
                "Where num_lib_dbb = " & num_libro
ARMA_PARTE_SOURCE_DE_LISTA_DE_CLIENTES = PSPARTE
If incluye_tipo6 Then Exit Function
'NO PASA SI INCLUYE CATEG 6
PSPARTE = PSPARTE & _
      " INTERSECT " & _
      "SELECT num_cliente " & _
      "From clientes " & _
      "Where clientes.categ_cli < 6"
ARMA_PARTE_SOURCE_DE_LISTA_DE_CLIENTES = PSPARTE
End Function
```

7.4 Conceptos básicos de índices de una base de datos relacional

Como se describió en el Capítulo 2, un índice es una estructura asociada con una tabla que acelera la recuperación de filas de la tabla misma, incluyendo el caso de una sola de ellas. Contiene claves generadas a partir de una o varias columnas de la tabla. Dichas claves están almacenadas en una estructura (árbol B) que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave.

Puesto que es uno de los temas que se analizaron en cuanto al desempeño de las consultas (y el espacio ocupado por la base de datos) se repite una explicación del concepto de un índice agrupado (clustered).

Índice Agrupado (Clustered)

Los índices agrupados ordenan y almacenan las filas de los datos de la tabla de acuerdo con los valores de la clave del índice. Son columnas incluidas en la definición del índice. Sólo puede haber un índice agrupado por cada tabla, porque las filas de datos sólo pueden estar ordenadas de una forma. La única ocasión en la que las filas de una tabla están ordenadas físicamente es cuando la tabla contiene un índice clúster. Cuando una tabla tiene un índice clúster, la tabla se denomina tabla agrupada. Si una tabla no tiene un índice clustered, sus filas de

datos están almacenadas en una estructura sin ordenar denominada montón (heap).

Índice No agrupado (non clustered)

Los índices no agrupados tienen una estructura separada de las filas de datos. Un índice no agrupado contiene los valores de clave de índice y cada entrada de valor de clave tiene un puntero a la fila de datos que contiene el valor clave.

El puntero de una fila de índice no agrupado hacia una fila de datos se denomina localizador de fila. La estructura del localizador de filas depende de si las páginas de datos están almacenadas en un montón o en una tabla agrupada. Si están en un montón, el localizador de filas es un puntero hacia la fila. Si están en una tabla agrupada, el localizador de fila es la clave de índice agrupado.

Impacto de la presencia de índices y del tipo de éstos

Para determinar el impacto de la presencia de algún índice sobre las duraciones de las consultas, se crearon 7 bases de datos, donde las diferencias entre ellas eran los índices incluidos en la tabla CLILIBRO. Las bases y sus respectivos índices se muestran en la Tabla 14. El concepto de un índice "clustered" se expone más abajo.

Tabla 14. Las bases de datos creadas para determinar el impacto de índices de la tabla VENTAS

BASE #	Libro y cliente	Cliente y libro	Cliente	Libro
1	**P-CL**		**S**	
2	**P-CL**			
3	**S**		**P-CL**	
4	**P**		**S**	
5		**P-CL**		
6			**P-CL**	
7		**P-CL**		**S**

P = índice principal, **P-CL** = índice principal clustered, **S** = índice secundario

La presencia y naturaleza de los índices afecta el plan de ejecución de las consultas. La sección 7.11 ilustra abundantemente este tema, usando diversas consultas y mostrando los respectivos planes de ejecución. Sin embargo, en los tiempos de respuesta obtenidos al efectuar consultas a las diferentes bases de datos, se incluyeron en el resto del capítulo.

Comentario: Por ser muy extensa, se postergó la sección que contiene los planes de ejecución para comodidad del lector que no está interesado en esos detalles.

7.5 Las consultas efectuadas y comentarios sobre las mismas

Se ejecutaron muchas decenas de miles de consultas, para las cuales se construyeron "lotes de consultas" que incluían diversos valores del parámetro principal de la consulta: el libro **L1**, de acuerdo al número de ventas del mismo. Estos lotes de consultas se ejecutaron variando

- la base de datos utilizada
- el orden en el que se efectuaban las consultas del lote.

Se decidió ejecutar las consultas de cada lote en 3 órdenes diferentes:

- en orden ascendente del número de ventas del libro
- en orden descendente
- iniciando con el libro del medio, ejecutando las consultas hacia arriba
- y luego las restantes hacia abajo. Supongamos que los libros del lote
- eran, L1, L2, L3, L4, L5, L6 y L7, en orden del volumen de ventas. La tercera ejecución ejecutaba las consultas en el orden l4, L3, L2, L1, L5, L6 Y L7.

La decisión de probar los 3 órdenes surgió de la gran variación de las duraciones de las mismas consultas cuando se las ejecutaba en diferentes circunstancias. Adicionalmente, se ejecutaron los lotes dos veces, y se registraron las duraciones de ambos: Se iniciaba una sesión (de Windows) y se ejecutaban dos veces los lotes. Esto se debió a que el SQL Server almacena todos sus elementos utilizados en memoria, incluyendo especialmente cualquier índice o tabla temporal que construya como parte de una consulta. De ese modo, la

repetición de un lote mostraba reducciones muy significativas de las duraciones, especialmente en el primer “libro”. El único modo de evitar que SQL Server usara datos que había adquirido o creado en consultas anteriores era reiniciar el servidor SQL Server.

Para registrar las duraciones resultantes, se diseñó una base de datos (en ACCESS) con una tabla que tenía los siguientes campos:

- cual-tecnologia as string ‘ SQL o DBB
- Cual-Base as integer
- número_de_cliente as long
- Primer_lote_de_la_sesión SI o NO
- El-orden as string (se ejecutaron los lotes en orden ascendente, descendente, del medio hacia arriba y luego abajo)
- duración en milisegundos as long (entero de 4 bytes).

Al final de las ejecuciones, se pasaron los datos a hojas de cálculo.

El lote de consultas seleccionado fue el que se muestra en la Tabla 15, que refleja que se variaron dos parámetros de una consulta a otra: el número de ventas del libro, y el cliente que lo adquirió, donde se tomaron clientes de diversas categorías. La primera hilera refleja una consulta “inútil” que se incluyó en muchos lotes para amortiguar el impacto de la primera consulta. Esto naturalmente disminuye los tiempos de respuesta del resto de consultas del lote, y se hizo para determinar estos tiempos de respuesta en las condiciones más favorables para el SQL.

Tabla 15. Libros seleccionados para las consultas

Libros y clientes seleccionados para las consultas 5 millones de clientes		
Ventas de ese libro	Libro número (**L1**)	CLIENTE (**C1**)
	9000	92289
20	9521	3125346
1064	313800	1542576
5001	317201	4933550
10601	318621	1861521
20001	320781	4562410
39951	320780	3212978

Cabe señalar que se probaron muchos otros lotes, es decir, con diferentes libros. Sin embargo, la relación entre los tiempos era aproximadamente la misma, de modo que se decidió adoptar este lote para documentar los tiempos de respuesta.

7.6 El programa con el que se ejecutaron las consultas

Se describe a continuación el programa que se utilizó para ejecutar las consultas, aunque sin detalles de código..El código fuente se ha incluido en el CD anexo de esta tesis.

Paso1

Mostrar una forma (ilustrada en la Figura 11) para que el usuario indique:

- cuál base de datos (el número)
- es o no el primer lote ejecutado en esa sesión de trabajo
- el orden en el que se ejecutan las consultas del lote
- se desea que el programa muestre el tiempo de respuesta para cada libro del lote.

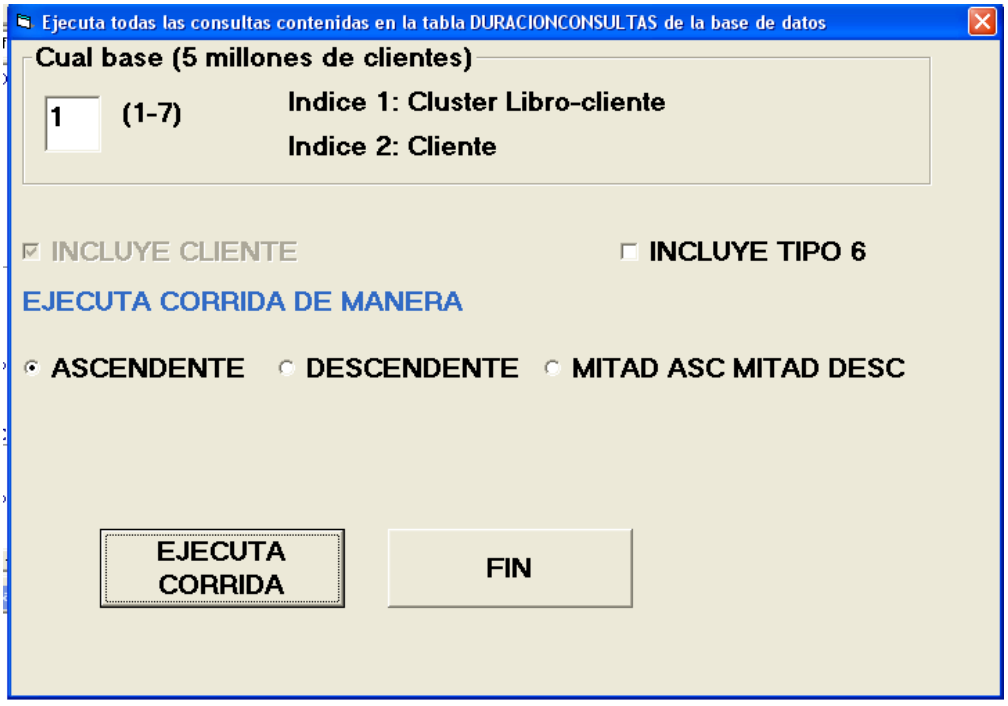


Figura 11. Pantalla para indicar parámetros y forma de ejecución de consultas.

Paso 2

- Conectar base de datos SOB apropiada
- Conectar base de datos de resultados apropiada para registro de duraciones.

Paso3

Para cada libro del lote

- Armar la consulta (con los 2 parámetros: número de libro y de cliente). Los valores se toman de la base de datos, que se prepara para dicho efecto.
- Ejecutar la consulta
- Registrar el tiempo de respuesta en la base de datos de resultados

La duración se mide a partir del momento en que se envía el comando a la base de datos (vía un Select) y el instante en que se termina el armado del Recordset resultante.

Paso 4

Si se seleccionó esa opción, desplegar tiempo de respuesta, que aparecen en una forma como la que ilustra la Figura 12. Naturalmente no se invoca esta opción cuando se ejecutan numerosas consultas en un mismo proceso.

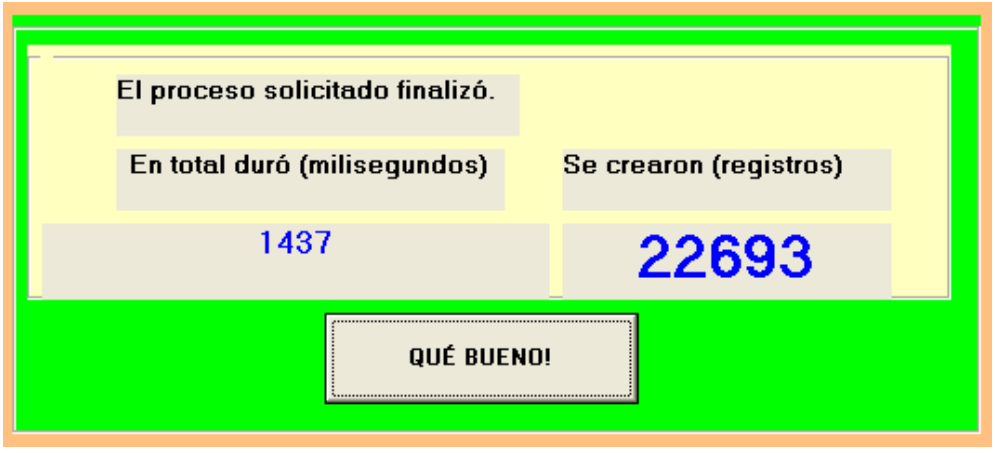


Figura 12. La forma en la que se muestran los resultados

7.7 Los tiempos de respuesta obtenidos

En la Tabla 16 se muestran los tiempos de respuesta obtenidos usando cada una de las bases que se indican. En cada celda se muestran los tiempos en los 3 órdenes mencionados, además del promedio de las 3 duraciones.

Tabla 16. Duraciones de las consultas efectuadas
(en segundos con dos dígitos decimales)

L1: Libro #		9521	313800	317201	318621	320781	320780
Ventas		20	1064	5001	10601	20001	39951
C1: Cliente		3125346	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro **C1**		43	10	386	7	66	43
Total libros en lista final		1497	4547	11490	22693	39078	73804
Base 1	Ascendente	3,56	2,13	1,45	3,11	5,39	14,95
	Descendente	0,03	0,30	0,63	1,20	3,11	4,31
	Asc-Descendente	0,03	0,28	0,64	1,16	2,95	4,39
	Promedio	**1,21**	**0,90**	**0,91**	**1,82**	**3,82**	**7,89**
Base 2	Ascendente	246,52	492,67	444,78	434,97	429,36	429,02
	Descendente						
	Asc-Descendente	232,69	437,83	453,47	435,27	436,69	435,88
	Promedio	**239,60**	**465,25**	**449,13**	**435,12**	**433,02**	**432,45**
Base 3	Ascendente	1,58	2,30	6,00	12,17	21,74	51,83
	Descendente	0,59	2,09	4,61	13,45	24,66	4,44
	Asc-Descendente	0,27	1,61	3,67	12,89	27,61	15,23
	Promedio	**0,81**	**2,00**	**4,76**	**12,84**	**24,67**	**23,83**
Base 4	Ascendente	3.63	2.79	1.81	3.33.	8.1	33.02
	Descendente	0.39	0.30	0.66	1.2	3.23	14.55
	Asc-Descendente	0.30	0.30	0.66	1.22	3.03	5.72
	Promedio	**1.44**	**1.13**	**1.04**	**1.92**	**4.76**	**17.76**
Base 5	Ascendente	405,91	367,69	363,33	363,81	364,08	362,49
	Descendente	370,66	388,74	375,45	374,44	387,33	430,36
	Asc-Descendente	381,56	392,83	385,78	372,78	361,00	360,08
	Promedio	**386,04**	**383,08**	**374,85**	**370,34**	**370,80**	**384,31**
Base 6	Ascendente	735.84	745.20	787.63	760.16	757.20	843.53
	Descendente						
	Asc-Descendente						
	Promedio	**735.84**	**745.20**	**787.63**	**760.16**	**757.20**	**843.53**
Base 7	Ascendente	0,64	0,92	1,73	2,78	7,84	21,36
	Descendente	0,11	0,31	1,16	3,08	7,45	20,16
	Asc-Descendente	0,03	0,28	0,72	1,44	7,05	18,42
	Promedio	**0,26**	**0,51**	**1,20**	**2,43**	**7,45**	**19,98**

Observación: El hecho de que ciertas hileras no contengan duraciones se debe a que no se realizaron las consultas correspondientes, puesto que las duraciones (al usar un índice no clustered) eran demasiado largas para ejecutar el lote en todos los órdenes.

En base a las duraciones reflejadas en la tabla anterior, se tomó una decisión: Sólo se usarán en las comparaciones las duraciones promedio, a pesar de que, como se ve, la primera consulta de cada tipo tarda considerablemente más que las siguientes, de modo que esta variación se pierde al tomar el promedio. De hecho, una de las ventajas que se obtuvieron con el uso del DBB (como se verá más adelante) es eliminar la duración exagerada de la primera consulta de un lote de éstas.

La tabla 17 muestra un resumen de la comparación de las consultas en las 7 bases, donde se reflejan los tiempos "promedio".

Tabla 17. Resumen de consultas en las 7 bases (sólo duraciones promedio)

L1: Libro #	9521	313800	317201	318621	320781	320780
Ventas	20	1064	5001	10601	20001	39951
Lista final	1497	4547	11490	22693	39078	73804
Base 1	1,21	0,90	**0,91**	**1,82**	**3,82**	**7,89**
Base 2	239,60	465,25	449,13	435,12	433,02	432,45
Base 3	0,81	2,00	4,76	12,84	24,67	23,83
Base 4	1.44	1.13	1.04	1.92	4.76	17.76
Base 5	386,04	383,08	374,85	370,34	370,80	384,31
Base 6	735.84	745.20	787.63	760.16	757.20	843.53
Base 7	**0,26**	**0,51**	1,20	2,43	7,45	19,98

La hilera "lista final" indica el número de libros resultantes, los que se ofrecerían al cliente C1. Observe que se resaltaron los "mínimos".

En la Tabla 18 se muestran los tiempos de respuesta obtenidos usando cada una de las bases que se indican, pero ahora excluyendo los clientes que compraron muchos libros (aquellos de categoría 6). En cada celda se muestran los tiempos en los 3 órdenes mencionados, además del promedio de las 3 duraciones.

Tabla 18. Duraciones en segundos y fracción de la consulta básica excluyendo los clientes de la categoría 6 en las 7 bases

L1: Libro #		9521	313800	317201	318621	320781	320780
Ventas		20	1064	5001	10601	20001	39951
C1: Cliente		3125346	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro **C1**		43	10	386	7	66	43
Total libros en lista final		603	1608	5501	11730	19608	39673
Base 1	Ascendente	1,50	2,86	1,42	3,09	5,95	12,42
	Descendente	0,00	0,19	0,44	0,70	1,77	2,83
	Asc-Descendente	0,02	0,17	0,39	0,72	1,78	2,91
	Promedio	**0,51**	**1,07**	**0,75**	**1,51**	**3,17**	**6,05**
Base 2	Ascendente	288,00	460,16	440,97	444,69	462,78	500,49
	Descendente	229,66	439,83	445,20	456,49	477,73	567,92
	Asc-Descendente						
	Promedio	**258,83**	**449,99**	**443,09**	**450,59**	**470,26**	**534,20**
Base 3	Ascendente	**1,89**	**5,22**	**3,56**	**8,33**	**16,45**	**28,22**
	Descendente	1,38	0,80	3,38	5,75	16,16	3,80
	Asc-Descendente	0,06	0,42	1,45	2,25	16,69	3,64
	Promedio	**1,11**	**2,15**	**2,80**	**5,44**	**16,43**	**11,89**
Base 4	Ascendente	2.29	2.72	1.38	2.92	6.1	28.01
	Descendente	0.29	0.21	0.47	1.18	2.73	11.88
	Asc-Descendente	0.28	0.2	0.44	1.15	2.52	6.22
	Promedio	0.95	1.04	0.76	1.75	3.78	15.37
Base 5	Ascendente	434,63	402,17	389,17	419,36	435,06	527,64
	Descendente	357,56	354,00	354,16	357,89	361,88	476,59
	Asc-Descendente	363,58	391,41	376,05	351,17	354,58	429,78
	Promedio	**385,26**	**382,53**	**373,13**	**376,14**	**383,84**	**478,01**
Base 6	Ascendente						
	Descendente						
	Asc-Descendente						
	Promedio						
Base 7	Ascendente	1,75	4,24	2,30	5,49	11,44	28,47
	Descendente	0,44	0,34	0,53	0,77	5,14	2,95
	Asc-Descendente	0,02	0,17	0,42	0,72	1,99	2,92
	Promedio	**0,73**	**1,58**	**1,08**	**2,32**	**6,19**	**11,45**

La tabla 19 muestra un resumen de la comparación de las consultas en las 7 bases, donde se reflejan los tiempos “promedio”.

Tabla 19. Duraciones promedio en segundos y fracción de la consulta básica en las 7 bases (excluyendo clientes de categoría 6)

L1: Libro #	9521	313800	317201	318621	320781	320780
Ventas	20	1064	5001	10601	20001	39951
C1: Cliente	3125346	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro **C1**	43	10	386	7	66	43
Total libros en lista final	603	1608	5501	11730	19608	39673
Base 1	**0,51**	**1,07**	**0,75**	**1,51**	**3,17**	**6,05**
Base 2	**258,83**	**449,99**	**443,09**	**450,59**	**470,26**	**534,20**
Base 3	**1,89**	**5,22**	**3,56**	**8,33**	**16,45**	**28,22**
Base 4	**0.95**	**1.04**	**0.76**	**1.75**	**3.78**	**15.37**
Base 5	**385,26**	**382,53**	**373,13**	**376,14**	**383,84**	**478,01**
Base 6	**No procesada**					
Base 7	**0,73**	**1,58**	**1,08**	**2,32**	**6,19**	**11,45**

7.10 Comentarios acerca de las duraciones en diversas bases

Estas duraciones reflejan uno de los grandes problemas que presenta el uso de BDR en este tipo de situaciones de análisis de negocios. Puesto que se formulan consultas imprevistas, es imposible planificar todos los índices que se necesitarán posteriormente, además de que no es conveniente dotar a una tabla de “todos” los índices por las implicaciones que esto tendría, tanto en el uso adicional de espacio en disco como en el incremento muy significativo de las duraciones de los procesos de actualización de datos.

En particular, si se usa un índice agrupado (clustered) los procesos de actualización cambian, puesto que no se reorganizan siempre (físicamente) los registros, sino que se maneja como si fuera un índice no agrupado, y con procesos posteriores se reordenan físicamente las filas. Esto se hace creando nuevamente todo el índice agrupado(Elmasri and Navathe, 1994). Para el resto de la tesis, sólo se presentan resultados obtenidos con la base denominadas Base 1, es decir, la de mejor desempeño, con los índices Libro_cliente (Principal y agrupado) & Cliente (secundario.)

Datos usados para la comparación con las respectivas duraciones en el modelo implementado en DBB (en el capítulo 10)

Todos los cuadros, como se mencionó anteriormente, se refieren a consultas efectuadas contra la Base # 1, y sólo incluyen las duraciones promedio.

La tabla 20 muestra un cuadro con las duraciones de la consulta incluyendo a los clientes de categoria 6

Tabla 20. Consulta básica (incluyendo cliente de la categoría 6)

L1: Libro #	9521	313800	317201	318621	320781	320780
Ventas	20	1064	5001	10601	20001	39951
C1: # Cliente	3125346	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro **C1**	43	10	386	7	66	43
Total libros en lista final	1497	4547	11490	22693	39078	73804
Duración	**1,21**	**0,90**	**0,91**	**1,82**	**3,82**	**7,89**

La tabla 21 muestra un cuadro con las duraciones de la consulta excluyendo a los clientes de categoria 6

Tabla 21. Consulta básica (excluyendo cliente de la categoría 6)

L1: Libro #	9521	313800	317201	318621	320781	320780
Ventas	20	1064	5001	10601	20001	39951
C1: # Cliente	3125346	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro **C1**	43	10	386	7	66	43
Total libros en lista final	603	1608	5501	11730	19608	39673
DURACIÓN	**0,51**	**1,07**	**0,75**	**1,51**	**3,17**	**6,05**

7. 11 Impacto de la presencia de índices en los planes de ejecución

Todo el material de esta sección, cuando menciona la presencia de un "ïndice" se refiere a un índice de la tabla "VENTAS" de la base de datos. Naturalmente, los planes también usan los índices de las otras tablas, en particular, LIBROS y CLIENTES; pero estas tablas tienen los mismos índices principales en todas las bases creadas.

Para conveniencia del lector de esta sección, se repiten en la tabla 22 las descripciones de las 4 bases que se usaron para ilustrar los planes de ejecución. Observe que se hicieron muchas consultas usando las otras 3 bases, pero no presentan diferencias interesantes en sus planes de ejecución y por lo tanto no fueron incluidas en esta sección.

Tabla 22. Índices de la tabla VENTAS de las bases de datos utilizadas para ilustrar los planes de ejecución

BASE #	Libro y cliente	Cliente y libro	Cliente	Libro
1	**P-CL**		**S**	
2	**P-CL**			
4	**P**		**S**	
5		**P-CL**		

P = índice principal, **P-CL** = índice principal clustered, **S** = índice secundario

Se incluyeron las consultas que refleja la Tabla 23, donde el número indicado se refleja en el número de subsección.

Tabla 23. Consultas para las cuales se muestran los planes de ejecución

	Base #	Consulta efectuada (criterios adicionales, además de la básica)
1	1	Ninguna
2	4	Ninguna
3	1	Sin "*" en el select de LIBROS (sólo se obtiene el número)
4	2	Ninguna
5	5	Ninguna
6	1	Excluye clientes de categoría 6
7	1	Excluye clientes de categoría 6 Y sin el "*" en select de LIBROS

A continuación se reproducen las sentencias SQL de las 4 consultas usadas en estos ejemplos.

Consulta básica (sin condiciones adicionales, con el Select *)

```
SELECT * From libros
WHERE num_lib_dbb
   IN(
      SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente
         IN{
              SELECT num_cliente From clilibro
               Where num_lib_dbb=L1
           }
      EXCEPT
      SELECT num_lib_dbb From clilibro
      Where num_cliente = C1
     )
```

Consulta básica (sin el Select *)

```
SELECT DISTINCT num_lib_dbb From clilibro
WHERE num_cliente
   IN{
        SELECT num_cliente From clilibro
         Where num_lib_dbb=L1
     }
EXCEPT
SELECT num_lib_dbb From clilibro
Where num_cliente = C1
```

Consulta básica (excluyendo clientes de categoría 6, con el Select *)

```
SELECT * From libros
   WHERE num_lib_dbb IN(
   SELECT DISTINCT num_lib_dbb From clilibro
       WHERE num_cliente IN (
     SELECT num_cliente From clilibro Where num_lib_dbb = L1
       INTERSECT
       SELECT num_cliente From clientes
       Where clientes.categ_cli < 6)
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1)
```

Consulta básica (excluyendo clientes de categoría 6, sin el Select *)

```
   SELECT DISTINCT num_lib_dbb From clilibro
       WHERE num_cliente IN (
     SELECT num_cliente From clilibro Where num_lib_dbb = L1
       INTERSECT
       SELECT num_cliente From clientes
       Where clientes.categ_cli < 6)
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1
```

7.11.1 Usa Base # 1, consulta básica sin condiciones adicionales

En la figura 13 se muestra el plan de ejecución de esta consulta.

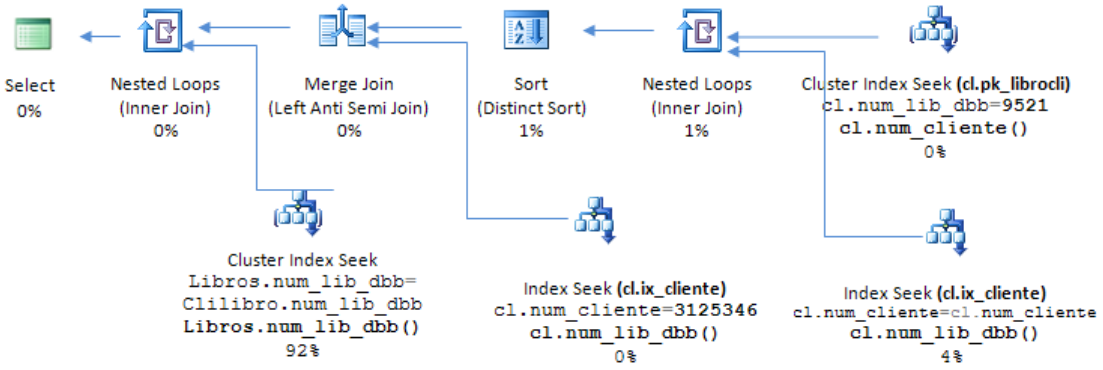


Figura 13. Plan de ejecución de la consulta basica con Base #1

7.11.2 Usa Base # 4, consulta básica sin condiciones adicionales

En la figura 14 se muestra el plan de ejecución de esta consulta.

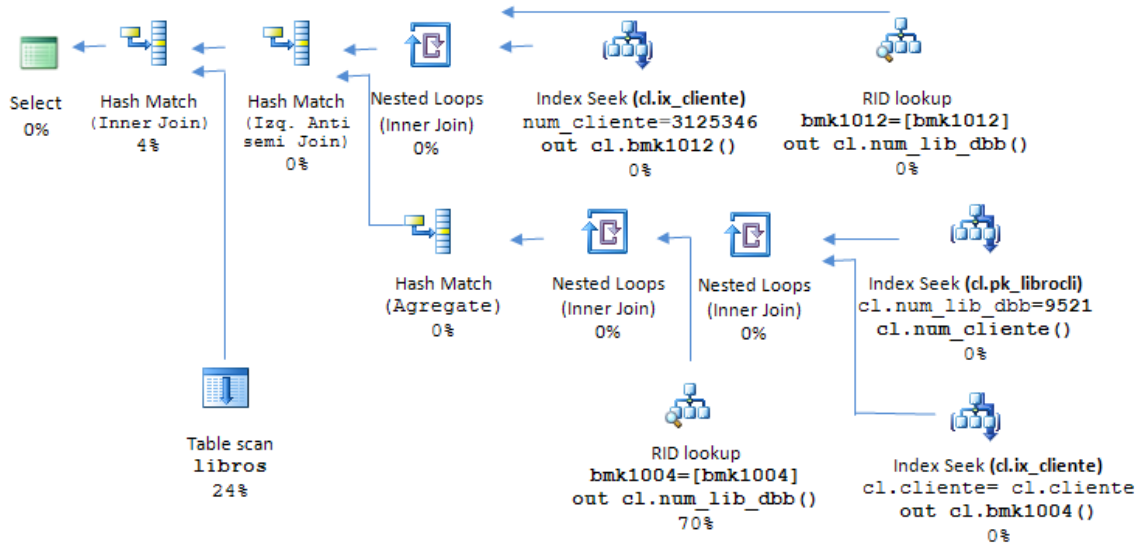


Figura 14. Plan de ejecución de la consulta basica con Base #4

7.11.3 Usa Base # 1, sin “*” en Select de LIBROS

En la figura 15 se muestra el plan de ejecución de esta consulta.

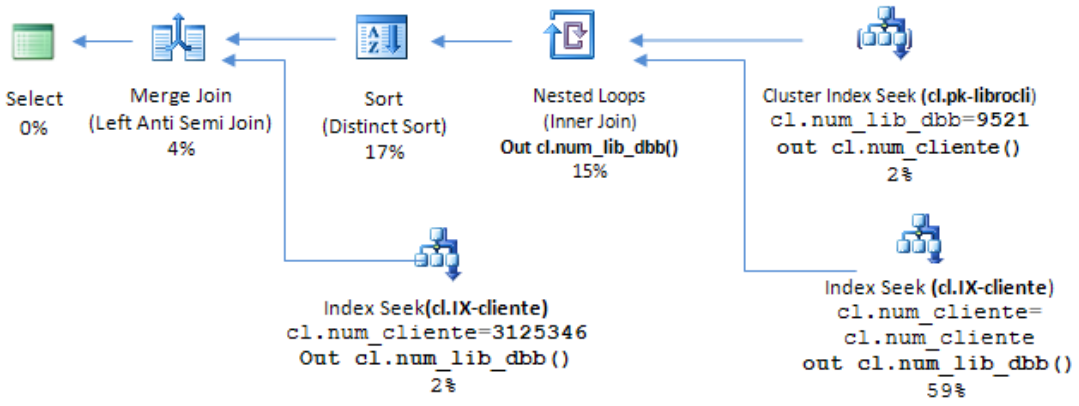


Figura 15. Plan de ejecución de la consulta basica con Base #1 sin el select “*”

7.11.3 Usa Base # 2, consulta básica sin condiciones adicionales

En la figura 16 se muestra el plan de ejecución de esta consulta.

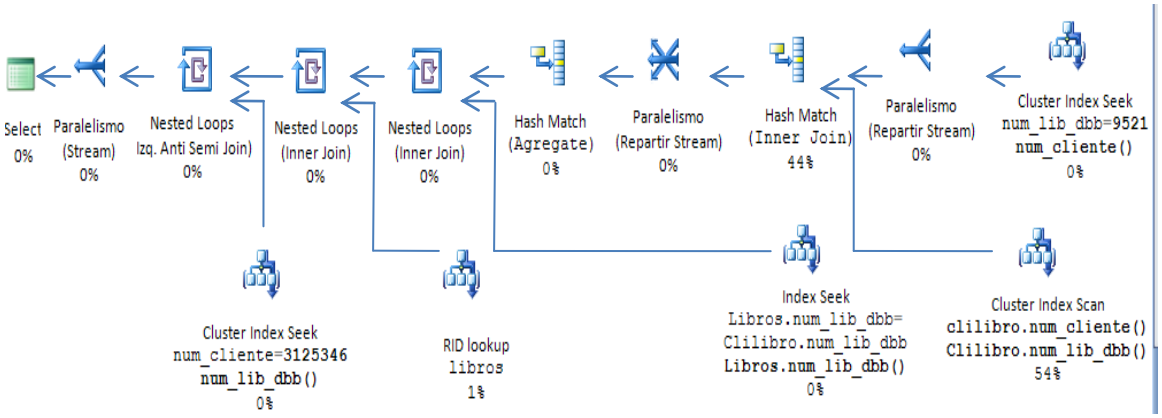


Figura 16. Plan de ejecución de la consulta basica con Base #2

7.11.4 Usa Base # 5, consulta básica sin condiciones adicionales

En la figura 17 se muestra el plan de ejecución de esta consulta.

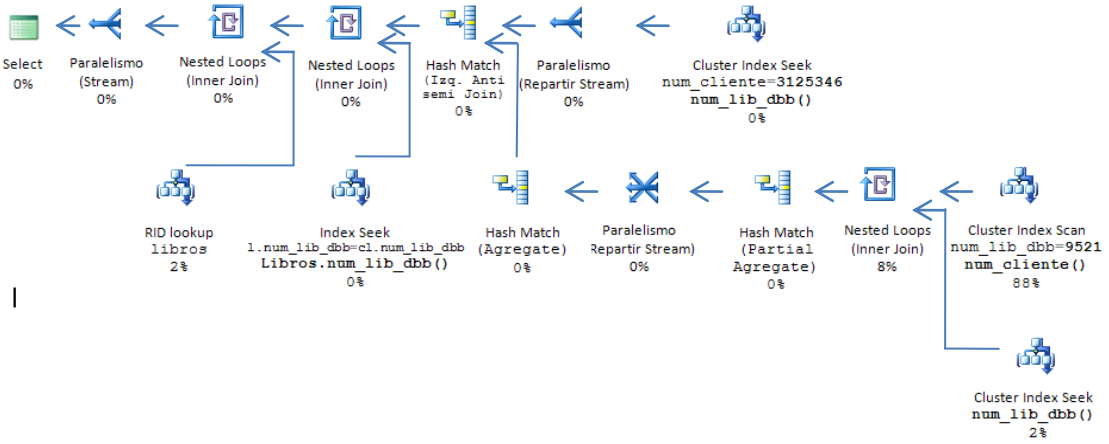


Figura 17. Plan de ejecución de la consulta basica con Base #5

7.11.5 Usa Base # 1, Excluye clientes de categoría 6

En la figura 18 se muestra el plan de ejecución de esta consulta.

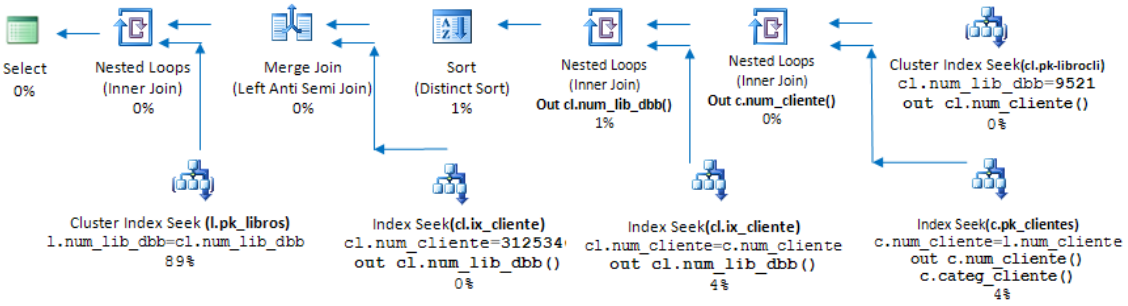


Figura 18. Plan de ejecución de la consulta basica con Base #5 excluyendo clientes de la categoría 6

7.11.6 Usa Base # 1, Excluye clientes de categoría 6 y sin el “*” en el Select

En la figura 19 se muestra el plan de ejecución de esta consulta.

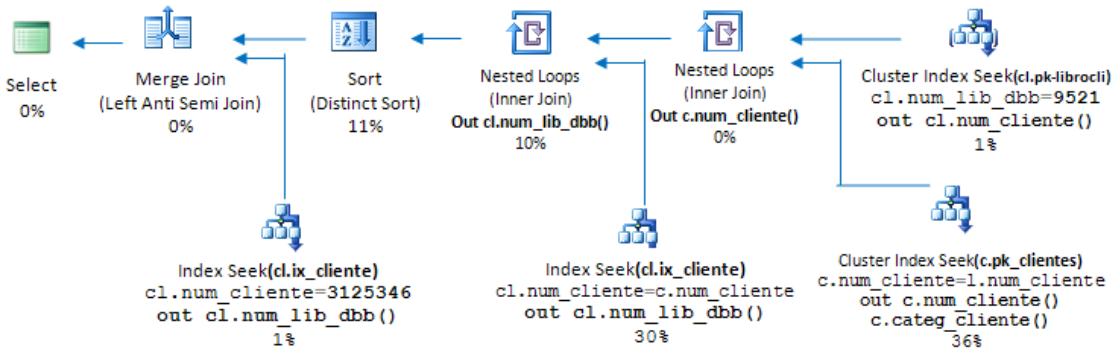


Figura 19. Plan de ejecución de la consulta basica con Base #1 excluyendo clientes de categoría 6 y sin el select *

CAPÍTULO 8. IMPLEMENTACIÓN DE LA SITUACIÓN DE NEGOCIOS EN DBB

Introducción

Al igual que en cualquier otra base de datos, se diseña una estructura para almacenar la información que se incluirá en la base. En una base de datos, esto se hace esencialmente en dos etapas:

- se definen las tablas de la base
- se definen los campos de cada tabla, y se agregan índices.

Posteriormente (o simultáneamente) se pueden agregar relaciones, procedimientos catalogados, vistas, funciones y disparadores.

En DBB, las actividades que se realizan como parte del diseño son un tanto distintas, aunque algunas se parecen a las que se hacen en una BDR.

En este capítulo se describen las actividades relacionadas con el diseño de la aplicación SOB usando el DBB. A continuación se describe el proceso ETL de carga de datos a esta estructura a partir de los datos generados para el SOB en la BDR.

Puesto que el DBB se basa en el paquete KBC para la gestión de las marcas, se intercalan observaciones sobre el uso del KBC, y en el caso de esta investigación, algunas modificaciones hechas a dicho paquete.

El diseño y la implementación de la aplicación SOB en la estructura DBB se presentan divididos en las siguientes secciones:

1. Seleccionar las opciones del DBB que se usarán en la aplicación
2. Definir los contextos que usará
3. Definir las clases de UBI
4. Construir una estructura de directorios para los datos
5. Preparar el uso del KBC
6. Carga de datos simulados (ETL) a la estructura definida.

8.1 Seleccionar las opciones del DBB que se usará la aplicación

El DBB es un paquete informático, y una aplicación es una instancia del paquete. La versión instalada del paquete mismo permite o no ciertas operaciones

y ofrece algunas de sus funciones. En particular, se especifican las opciones de la versión instalada, mismas que no pueden ser cambiadas por los usuarios del paquete. Esto permite la instalación de programas que no contienen elementos que no se necesitan en ciertas instalaciones. Para citar un ejemplo, puede ser que todos los que usen el paquete lo harán en el mismo idioma, de modo que ofrecer otros carecería de sentido.

Como se comentó en el capítulo 3, la exclusión de alguna opción tiene como objetivo el que los programas de actualización, tanto en la etapa de diseño del esquema del DBB como los que se usan para actualizar los datos, no ofrezcan funciones u opciones que los usuarios no usarán. Sin embargo para este proyecto (el SOB) se elaboró un programa “incompleto” del DBB, precisamente con las opciones que se señalan en la tabla 24.

Tabla 24. Las opciones del paquete DBB que se implementaron

1	ofrece uso multi-usuario del DBB	**NO**
2	ofrece selección de rdbms para instancias	**NO**
3	usa directorio de "paths" (directorios)	**SI**
4	Ofrece uso de marcas virtuales	**NO**
5	Ofrece cambios masivos	**NO**
6	ofrece marcado masivo de fichas existentes	**NO**
7	ofrece traducción de nombres de objetos del DBB	**NO**

Observación. Las 7 opciones que se muestran no constituyen la totalidad de éstas que contempla el DBB. No se describen las restantes opciones puesto que no tienen relación con la aplicación SOB.

A continuación se seleccionaron las opciones que se implementarían en la versión de los programas que se usarían para efectuar la carga y el uso de los datos. En la Tabla 25 se muestran las opciones las opciones que se consideraron para el programa.

Tabla 25. Opciones de la instancia SOB

1	usa clases (0= una clase única)	**NO**
2	Traducción "inmediata" nombres de objetos de clases	**NO**
3	usa marca 901 (todas las fichas de la instancia)	**NO**
4	usa marca 902 (las fichas de una clase)	**NO**
5	usa marcas 701, 702, ETC palabras en orden	**SI**
6	ofrece listas D (varias listas de valores por registro)	**NO**
7	ofrece listas I (intervalos de bitmaps)	**NO**
8	ofrece consultas avanzadas	**SI**
9	ofrece consultas catalogadas	**NO**
10	ofrece crea base resultados a partir de listas [SI]	**NO**
11	ofrece catálogo de bases datos de resultados	**NO**
12	ofrece filtros basados en campos adicionales [no]	**SI**
13	ofrece operaciones muy "caras" [SI]	**NO**
14	ofrece importación de fichas de otras fuentes	**SI**
15	ofrece incorporación de datos otras fuentes a fichas existentes	**NO**
16	Ofrece usuarios con roles diferentes para diversas clases	**NO**
17	Usa grupos de contextos	**NO**

Para las clases de una instancia, además de las opciones heredadas (del paquete DBB mismo y de la aplicación) se pueden seleccionar de las opciones que se muestran en la Tabla 26, las necesarias para las clases que se definan en la aplicación.

Tabla 26. Opciones relacionadas con las clases

1	usa billón único (1) o varios billones (0)	**SI**
2	Usa numeración especial para fichas de la clase	**SI**
3	ofrece los 2 tipos de texto (1 = si)	**NO**
4	ofrece marcas aisladas	**SI**
5	ofrece materiales asociados a fichas	**NO**
6	Usa grupos de campos adicionales	**NO**
7	Usa campos adicionales	**SI**
8	ofrece campos tipo "memo"	**NO**
9	Usa vectores	**SI**
10	Usa vectores de dimensión variable	**SI**
11	Usa estructuras de vectores (vectores paralelos)	**SI**
12	Usa reglas para nomenclatura de elementos de vectores	**NO**
13	Usa “idiomas”	**NO**
14	Usa filtros basados en campos adicionales o vectores	**NO**
15	ofrece formatos de campos en las clases	**NO**
16	ofrece opciones confidencialidad a nivel grupos	**NO**
17	Ofrece marcas virtuales	**SI**
18	Nivel de auditoria de campos de la ficha	**NO**
19	Ofrece multi-valor una marca	**SI**
20	ofrece_multi-contexto_un_valor (palabras)	**NO**
21	nivel de deformación de datos de fichas	**NO**

Se explican a continuación algunas de las opciones cuyo significado o impacto pudiera no estar muy claro solamente por su nombre. Observe que se han indicado los números para referencia con la tabla anterior.

1 Usa billón único (1) o varios billones (0): El DBB está basado en enteros de 4 bytes. Por ejemplo, el número de UBI es un tal entero. Esto limita el número de fichas de un acervo. Para aumentar la capacidad, se asigna el número de ficha separado: el billón (Un entero de 2 bytes) y el número, que indica los últimos 9 dígitos del nombre. Observe que el término “billón” se refiere a mil millones (es decir, la terminología inglesa) y no a un millón de millones, como en español.

En SOB no se excedió nunca el número 999,999,999 en ninguno de los números, ni en las fichas del DBB ni en el uso del KBC.

2 Usa numeración especial para fichas de la clase: se pueden usar rangos de números para las clases (o no.) En el caso del SOB, se reservó el intervalo 1 a

10.000.000 para los números de los libros, mientras que las fichas de clientes tienen números del 10.000.001 en adelante.

3 Usa grupos de campos adicionales: como se describió en el Capítulo 3, el DBB permite agrupar los campos adicionales de una clase en grupos. En SOB no fue necesario esta agrupación, por dos motivos:

- no se protegieron campos contra uso no autorizado
- las clases no tienen un número elevado de campos adicionales.

12 Usa reglas para nomenclatura de elementos de vectores: en DBB se pueden incluir reglas para que, al mostrar los elementos de un vector, se les asocie un nombre. Por ejemplo, si se guardan datos mensuales como elementos de un vector, aparecería el nombre del mes cuando se usen los datos.

14 Usa filtros basados en campos adicionales o vectores: como se explicó antes, a las listas de resultados obtenidas por consultas basadas en los contextos, se les pueden aplicar filtros basados en los campos fijos. Si se desea ampliar esta posibilidad para incluir criterios de selección basados en otros campos, se indica que las clases podrán ofrecer este tipo de filtros.

15 Ofrece formatos de campos en las clases: se pueden incluir “formatos” asociados a los campos de una ficha, para que se muestren de ese modo en las consultas. Por ejemplo, se puede indicar que se incluya el símbolo de “$” cuando se muestre un importe.

16 Ofrece opciones confidencialidad a nivel grupos: se pueden proteger los campos de un grupo. Si se trata de negar el uso (ya sea en actualización o sólo en consulta) el DBB protegerá los campos del grupo cuando un usuario no tenga la autorización necesaria para usarlos.

17 Ofrece marcas virtuales: una marca virtual consiste en un par (contexto, valor) en una ficha, pero donde el valor no está reflejado en un campo de la ficha.

18 Nivel de auditoria de campos de la ficha: el DBB ofrece varios niveles que resultan en diversos niveles de protección de las fichas para protegerlas contra alteraciones no autorizadas, es decir, que se efectúan sin usar los programas del DBB.

19 Ofrece multi-valor una marca: esto significa que puede haber varios valores (en una misma ficha) que se marcan con el mismo contexto. Esto es el caso cuando se marcan vectores con un contexto (se envía a KBC una marca para cada elemento del vector.)Pero también puede haber varios valores de una misma ficha que se marcan con el mismo contexto (ya sea campos adicionales, marcas aisladas, o palabras del texto marcable.)

20 Ofrece_multi-contexto_un_valor (palabras): éste es el caso de los autores de un libro en el SOB. Se guarda el nombre de cada autor, pero se marcan en forma individual las palabras que componen el nombre. El DBB reserva las marcas 720 en adelante para estas marcas.

21 Nivel de deformación de datos de fichas: el DBB ofrece la posibilidad de deformas total o parcialmente las fichas. Por ejemplo, se puede solicitar que los textos se almacenen en forma deformada, para impedir su lectura sin el uso de los programas correspondientes.

8.2 Definir los contextos que usará

A continuación se definen los contextos que necesita la aplicación, y los atributos de los mismos. En la Tabla 27 se muestran estos contextos. Observe que los contextos tipo “catálogo” son del tipo 4: permiten introducir valores sin instancia, que hacen las veces de catálogos. Por ejemplo, el KBC rechazará marcas del contexto 2 (categoría del cliente) con algún valor diferente a los válidos (1,2,3,4,5,6).

Tabla 27.Los contextos definidos para la aplicación SOB

#	NOMBRE	ATRIBUTO	VALORES POR CONTEXTO	INSTANCIAS POR VALOR
1	TIPO DE LIBRO	CATÁLOGO	6	100000
2	CATEGORÍA DEL CLIENTE	CATÁLOGO	6	1500000
3	TEMA	CATÁLOGO	16	20000
4	SUBTEMA **	NORMAL		
5	EDITOR **	NORMAL		
6	PALABRA CLAVE	CATÁLOGO	400	INDEFINIDO
7	IDIOMA ORIGINAL	CATÁLOGO	9	100000
8	OTRO IDIOMA	CATÁLOGO	9	40000
9	AÑO DE PUBLICACIÓN	CATÁLOGO	8	40000
10	PAÍS	CATÁLOGO	10	500000
11	CIUDAD	CATÁLOGO	100	50000
12	CLIENTE QUE COMPRÓ LIBRO	NORMAL	300000	30000
15	CODIGO POSTAL **	NORMAL		
701	1ERA PALABRA CONSECUTIVA	NORMAL	INDEFINIDO	
702	2DA PALABRA CONSECUTIVA	NORMAL	INDEFINIDO	
721	CAMPO SI-NO (1)	SOLO VALOR 1	INDEFINIDO	INDEFINIDO
722	CAMPO SI-NO (2)	SOLO VALOR 1	INDEFINIDO	INDEFINIDO
901	LA CLASE DE LA UBI **	CATALOGO		

Observación: los contextos marcados con ** no se actualizaron para la simulación. Es decir, no se generaron las marcas correspondientes, puesto que no se usarían en las consultas planeadas para la comparación. En particular, no se usó el contexto 901 puesto que las clases se numeraron usando rangos disjuntos de números, de modo que no fue necesario poder usar el contexto 901 para limitar una lista de resultados a las fichas de una clase.

Los valores por contexto e instancias por valor representan una estimación del número máximo de estos datos. Se documentaron aquí pero no se le pasaron al KBC puesto que no los necesitaba: para todos los contextos usó estructuras “L” para las listas de instancias de valores, es decir, en el árbol de valores se incluye el número de instancias y un apuntador al inicio de la lista en un archivo, y en el archivo se graba un arreglo de las instancias del valor.

8.3 Definir las clases de UBI

En base a los datos que hay que guardar, y el modo de usarlos, se determinó que el SOB usaría dos clases de fichas.

- Clase 1 LIBROS cada libro tendrá una ficha de esta clase.

- Clase 2 CLIENTES cada cliente tendrá una ficha de esta clase.

A continuación, se definen las clases, actividad que se efectúa en dos grupos:

Los datos de la clase misma

- especificar las opciones para la clase (las que usa)
- si procede, indicar el rango de numeración que se reserva para la clase
- indicar si se desea o no el marcado de fichas de la clase (contexto 901)
- indicar si usará´”marcas aisladas”
- indicar si se usarán materiales asociados, y si procede, indicar el tipo de éstos
- indicar si ofrece deformación de campos (scrambling)
- indicar cuáles contextos usa la clase.

Los campos de las fichas

- indicar el uso de los campos fijos
- indicar si se usarán textos (marcables y no marcables)
- definir los grupos de campos
- definir los campos adicionales (campos, vectores, estructuras)
- si hay estructuras, indicar cuáles vectores la componen
- indicar los marcados automáticos de campos fijos y adicionales
- indicar los niveles de deformación de campos (si procede)
- definir las marcas calculadas.

8.3.1 La clase CLIENTES

Para ilustrar el proceso de definición de una clase, se muestran algunas de las interfaces con las que se especifican los datos. No se muestran todas las formas, y algunas se han modificado *ad hoc* para efectos de su inclusión en esta tesis, puesto que las reales tienen muchos elementos “superpuestos” (que aparecen y desaparecen). Además, se han agrandado las “fuentes” para permitir la lectura en las figuras que muestran las formas.

Es conveniente tener claro lo que se va a incluir en la clase antes de introducirlo, pero el programa permite cambiar cualquier elemento de la clase. Sin

embargo, si la clase *ya tuviera fichas*, este proceso (por ahora) está deshabilitado. Habrá un programa del DBB que convierte las fichas “anteriores” para incorporar los nuevos datos, pero dicho programa no está elaborado todavía.

En la Tabla 28 se muestra cómo se almacenaron los datos de los clientes en las UBI’s de esta clase. Para los campos fijos, se indica su uso en esta clase, mientras que para los restantes campos, se usa una descripción inversa: se indica lo que se desea guardar y a continuación, qué tipo de campo de la ficha se usa, a pesar de que en la tabla se ilustran “al revés”.

Tabla 28. Los campos de una UBI de la clase CLIENTE

CAMPO DE DBB	USO EN LA CLASE CLIENTE	CONTEXTO
LOS CAMPOS FIJOS		
Número de UBI	Número de Cliente (+ 10000000)	
Billión	No se ocupa en SOB	
Clase	2 (clients)	
Título	Nombre del cliente	
Status	No se usa (siempre vale 1 en SOB	
Entero 1-byte	Categoría del cliente	**2**
Entero 2-bytes-1	Código postal	**15****
Entero 2-bytes-2	Tarjeta de Crédito- Tipo	
Entero 4 bytes	Cuántos libros compó	
Texto (16)	Nümero de tarjeta de crédito	
Fecha	Año y mes de primera compra	
Fecha 2-bytes	Año mes de última compra	
Clasif-1		
Clasif-2		
16 campos SI-NO	No se usan	
OTROS DATOS DE LA UBI		
Texto no marcable	Pregunta (autenticación)	
Texto no marcable	Respuesta	
Texto no marcable	Correo electrónico	
Texto no marcable	Otro correo electrónico	
Texto no marcable	Domicilio	
Texto no marcable	Teléfonos	
Texto no marcable	Banco de la tarjeta de crédito	
Virtual	Primera palabra del nombre	**701**
Virtual	Segunda palabra del nombre	**702**
Contexto-valor	País	**10**
Contexto-valor	Ciudad	**11**
Estructura de vectores		
Vector 1 (dim. variable)	Números de los libros comprados	**12**
Vector 2 (dim. variable)	Año y mes de la compra	
Campo auditoría 1	Palabra clave del cliente	

Observaciones

- El contexto 15 (código postal) no se incluyó en la simulación.
- La palabra clave del cliente se almacena como un entero de 4 bytes (calculado a partir de la palabra clave misma) en uno de los campos de auditoría de la UBI.
- Los valores del vector que contiene los libros comprados por el cliente se marcan con el contexto 12 (cliente que compró el libro). El valor del contexto es el número de la UBI correspondiente al libro (el elemento del vector) y la instancia es el número del cliente.

Al invocar la creación de una clase nueva aparece una forma como la que se muestra en la Figura 20. Si se copia una clase existente, se crea una clase idéntica (en todo sentido) a la que se usa como fuente, y luego se pueden cambiar opciones, campos, etc.

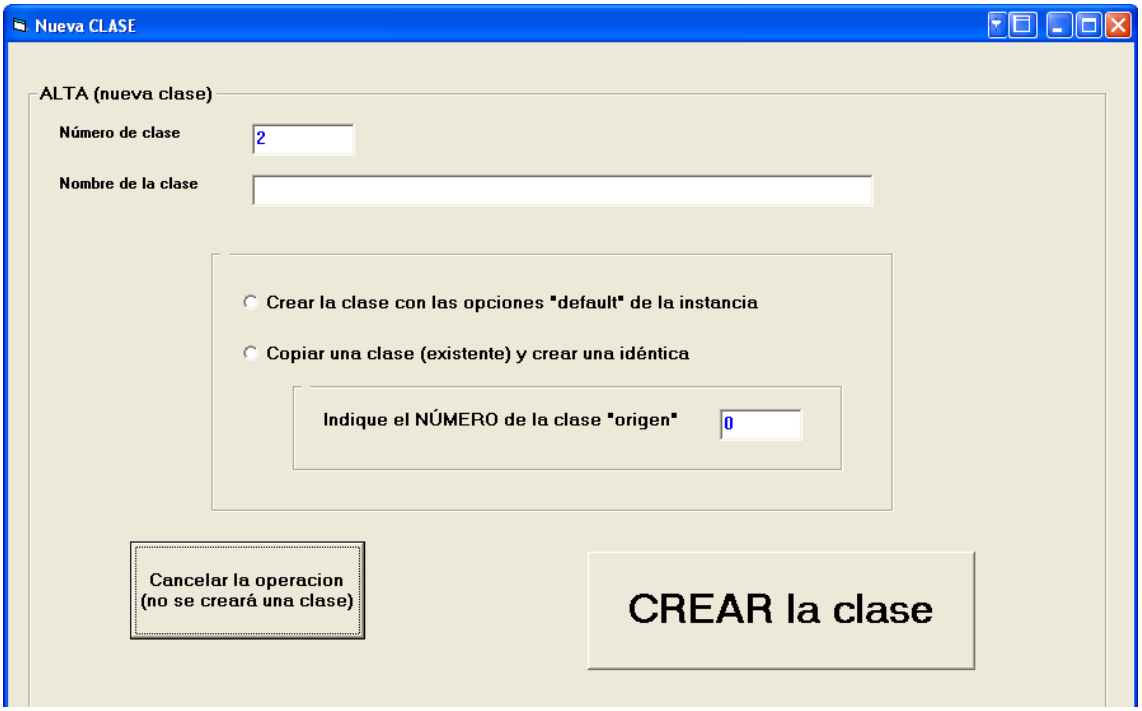
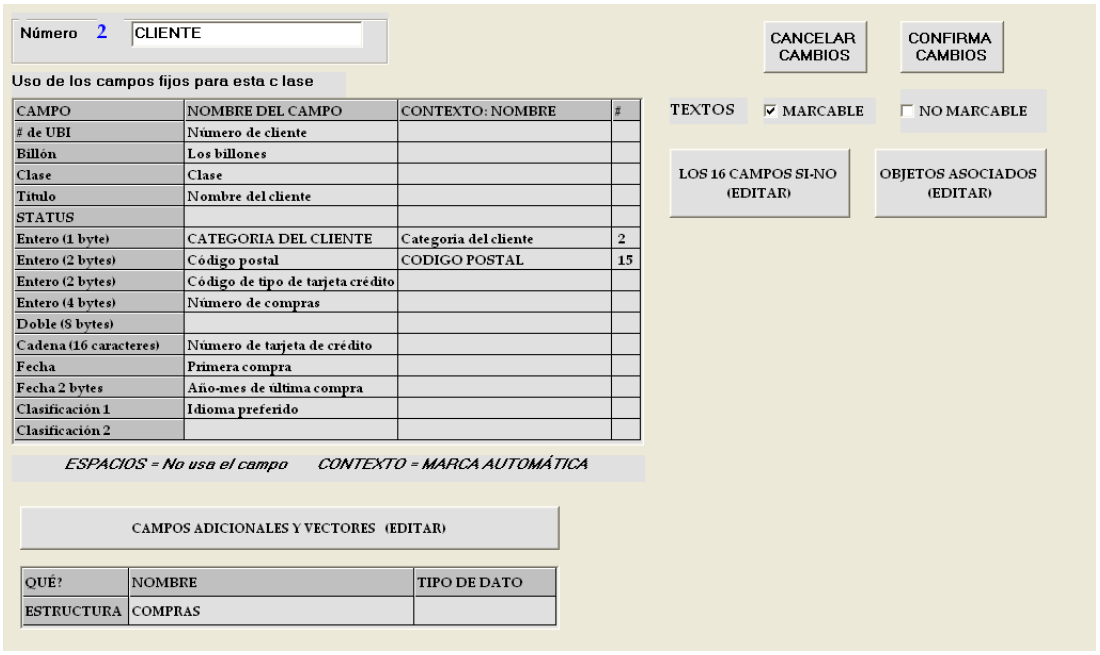


Figura 20. La forma para crear una clase

A continuación se actualizan los campos de la clase. La figura 21 ilustra la forma con la cual el encargado de hacerlo introduce estos campos.



CAMPO	NOMBRE DEL CAMPO	CONTEXTO: NOMBRE	#
# de UBI	Número de cliente		
Billón	Los billones		
Clase	Clase		
Titulo	Nombre del cliente		
STATUS			
Entero (1 byte)	CATEGORIA DEL CLIENTE	Categoria del cliente	2
Entero (2 bytes)	Código postal	CODIGO POSTAL	15
Entero (2 bytes)	Código de tipo de tarjeta crédito		
Entero (4 bytes)	Número de compras		
Doble (8 bytes)			
Cadena (16 caracteres)	Número de tarjeta de crédito		
Fecha	Primera compra		
Fecha 2 bytes	Año-mes de última compra		
Clasificación 1	Idioma preferido		
Clasificación 2			

QUÉ?	NOMBRE	TIPO DE DATO
ESTRUCTURA	COMPRAS	

Figura 21. La definición de la clase # 2: CLIENTE

Al seleccionar un campo fijo con un “click” en la hilera correspondiente (en este caso la hilera correspondiente al Entero de 1 byte) se agrega una función (como se ilustra en la Figura 22) en la cual se pueden modificar los valores para ese campo (el nombre y, si procede, el número de contexto con el que se marca este campo.) Observe que el botón identificado con un “?” ocasiona que se muestre una lista de los contextos definidos para la aplicación para que se seleccione el apropiado.

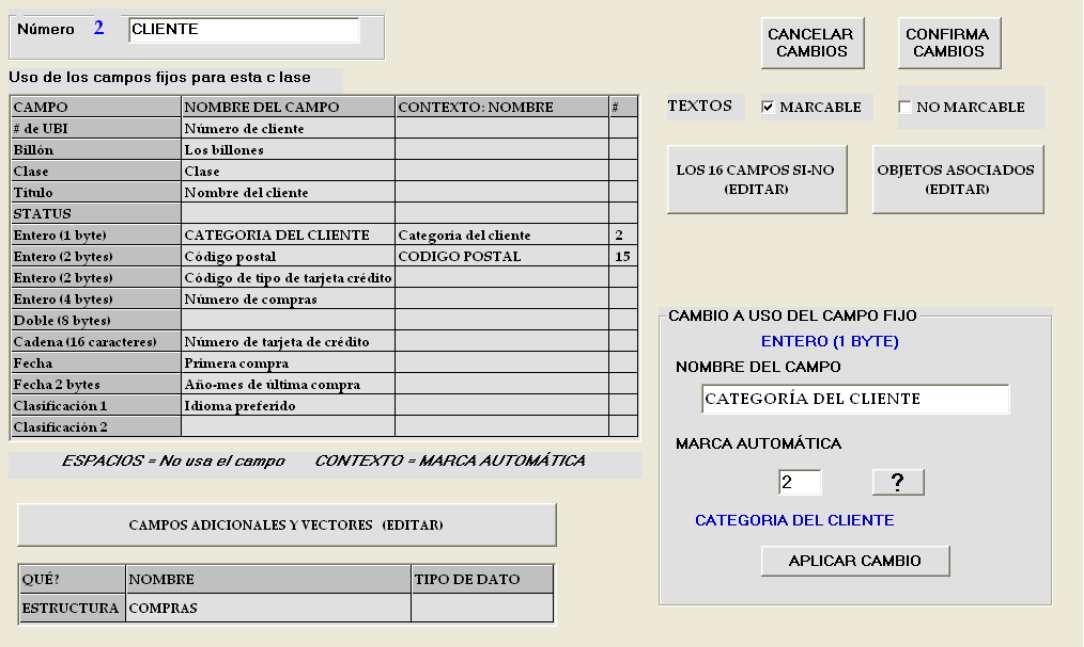


Figura 22. Cambios al uso de un campo fijo de la clase

Se indica si la clase usa o no textos marcables y no marcables.

El botón de edición de los 16 campos si-o-no invoca una forma que se ilustra en la figura 23, en la cual se indica el uso de estos campos. Estos campos siempre serán marcados con el contexto 720 + el subíndice del campo (1 a 16.) Estos contextos son parte de la definición del DBB, y sólo enviarán al KBC las instancias en las que los campos tienen un valor “1”.

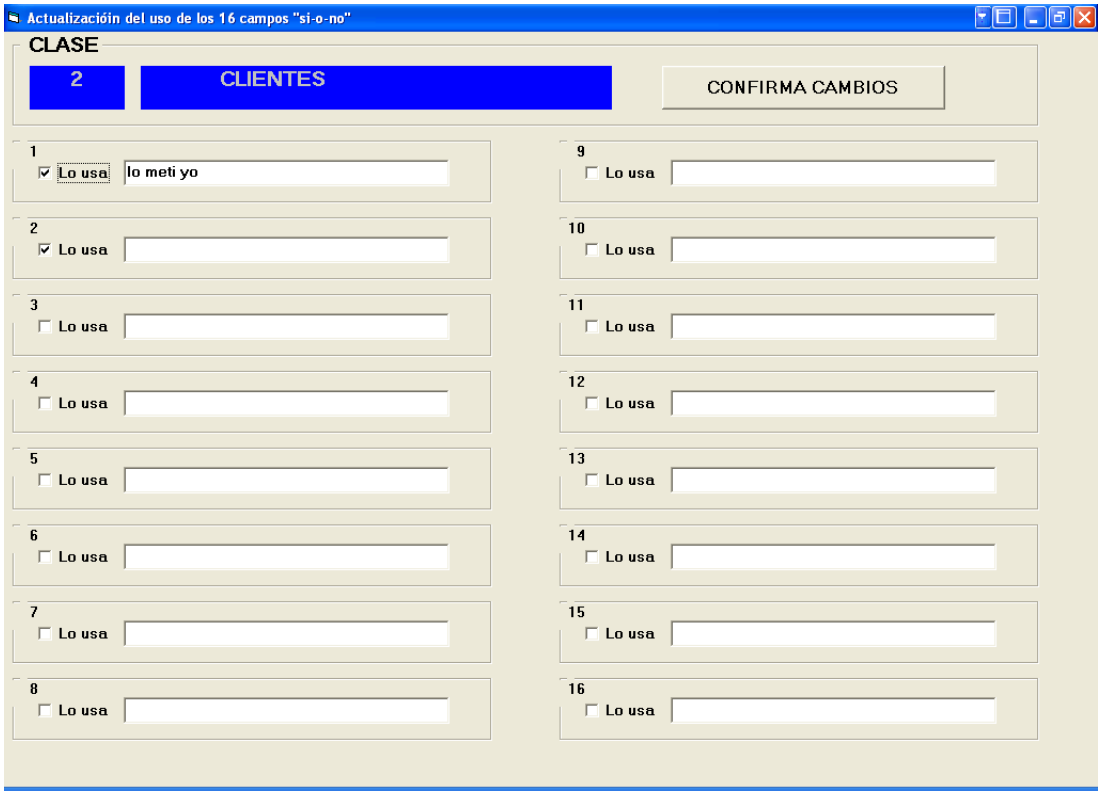


Figura 23. Cambios al uso de los 16 campos Si-o-NO de la clase.

El botón de edición de materiales adicionales permite indicar qué tipo de materiales permitirá la clase. La figura 24 ilustra la forma con la que se efectúa esta operación.

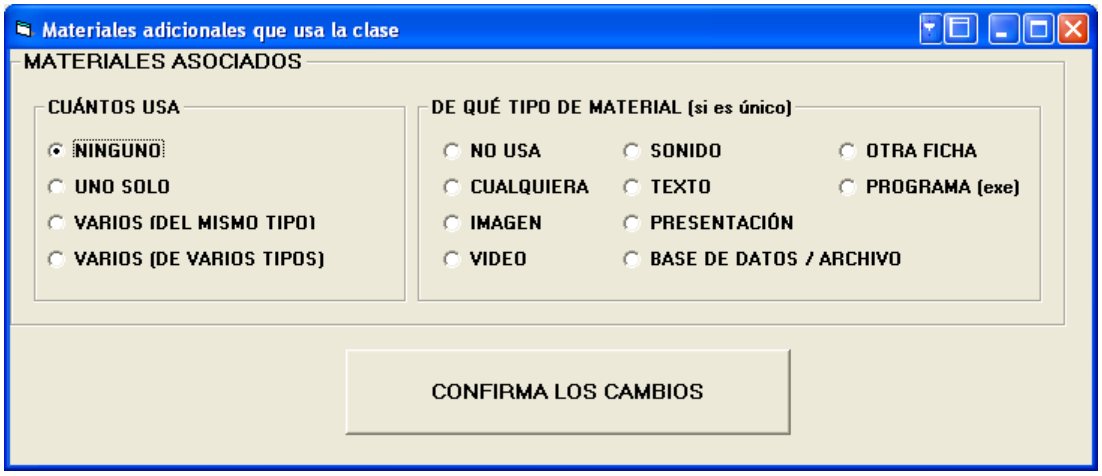


Figura 24. Forma para indicar el tipo de materiales asociados de una clase

La clase CLIENTES usa una Estructura (de vectores paralelos). No tiene otros campos ni vectores adicionales.

Al invocar la edición de campos adicionales aparece otra forma (que no se ilustra) donde se pueden agregar, eliminar o modificar los campos. Cabe señalar que los campos tienen diversos atributos, además del nombre y el tipo de dato.

Para las estructuras, se modifican éstas indicando los vectores que la componen, y la dimensión de los mismos (que puede ser variable). Todos los vectores de una estructura tienen la misma dimensión, puesto que los valores que se indicarán en sus elementos corresponden a un mismo dato.

En el caso de la clase clientes, la estructura tiene dos vectores de dimensión variable: 1 – Número de libro, como entero de 4 bytes, 2 – Año y mes de la compra (como entero de 2 bytes).

8.3.2 La clase LIBROS

Se crea una UBI (ficha) para cada libro. Los campos de la UBI se muestran en la tabla 29.

Tabla 29. Descripción de una UBI de la clase LIBRO

CAMPO DE DBB	USO EN LA CLASE CLIENTE	CONTEXTO
LOS CAMPOS FIJOS		
Número de UBI	Número de Libro	
Billion	No se ocupa en SOB	
Clase	1 (libro)	**901**
Título	Título del libro	
Status	No se usa (siempre vale 1 en SOB	
Entero 1-byte	Tipo de libro (según número de ventas)	**1**
Entero 2-bytes-1	Año de publicación	**9**
Entero 2-bytes-2	Tema	**3**
Entero 4 bytes	Cúantos ejemplares vendidos	
Texto (16)	Editor	**5 ****
Fecha		
Fecha 2-bytes		
Clasif-1	Idioma original	**7**
Clasif-2		
16 campos SI-NO	(1) Todavía se ofrece	**721 ****
	(2) Hay en existencia	**722 ****
OTROS DATOS DE LA UBI		
Texto no marcable	Resumen (abstract)	
Contexto-valor	Palabras clave	**6**
Contexto-valor	Subtema	**4 ****
Contexto-valor	Idiomas adicionales	**8**
VECTORES		
VDV (Texto-24)	Autores del libro (se marcan 2 palabras)	**701, 702**

Observación: los contextos marcados con ** no se incluyeron en la simulación.

Sólo se marcan las 2 primeras palabras de cada autor, con los contextos consecutivos 701 y 702. Estas marcas se usan en forma conjunta e indican que las palabras marcadas con los contextos 701 y 702 (y los subsecuentes si hubiera más de 2 palabras) aparecen en ese orden en el texto marcado (en este caso, el nombre del autor.) El contexto 721 se refiere al primero de los 16 campos SI-o_NO, el 722 al segundo, y así sucesivamente (si se usaran otros de estos campos.)

8.4 Construir una estructura de directorios para los datos

Se creó un directorio – EL SOB - para todos los archivos, con varios subdirectorios. Las bases de datos descritas en capítulos anteriores no se integraron a estas carpetas, sino se almacenaron de acuerdo a las especificaciones del SQL Server, y en diversos discos (de diversas computadoras). En la tabla 30 se muestran los subdirectorios y los el tipo de archivos que se almacenan en cada uno de ellos.

Tabla 29. Subdirectorios del directorio EL SOB y lo que contienen

Subdirectorio	Subdirectorio	Contenido (cuáles archivos)
RAÍZ		Programas ejecutables La DLL del KBC
SISTEMA	PRSOB	Programas del SOB ➢ Ejecuta consultas en SQL ➢ Prepara todo lo del SOB en DBB ➢ Carga datos de SQL Server ➢ Ejecuta consultas en DBB y registra en base de datos de duraciones
	PRGENERADATOS	Programa que genera datos en base SQL Server
SOBFILES	FICHAS	Fichas (UBI's) del SOB en DBB
	TEMPORALES	Diversos archivos intermedios
	LISTAS DE RESULADOS	Listas de resultados "temporales" que usan las consultas
KBCFILES	DIVERSAS	No se detallan las carpetas que usa el kbc.
DURACIONES		Bases de datos con duraciones de consultas

KBCFILES es el directorio que usa el paquete KBC. Cuando se instancia el paquete, se le informa dónde debe ir este directorio (en este caso, c:\EL SOB". Como el KBC usa muchos tipos de archivos, los divide en varios directorios, mismos que no crea el usuario sino el paquete mismo.

8.5 Preparar el uso del paquete KBC

La siguiente etapa de la preparación de un "DBB" *ad hoc* para el SOB fue la revisión y pruebas del paquete KBC, con las opciones necesarias para este proyecto.

Al inicio del proyecto SOB, había una versión funcional del KBC. El KBC usa árboles B (o B+, como se los llamaba antes) para guardar tanto los valores de un

contexto como las instancias de cada valor (ya se había mencionado que estas estructuras caen dentro del concepto de un GIN.) En la versión disponible, se usaron tablas de una base de datos en ACCESS para implementar los árboles, es decir, se usaron los índices de la base de datos relacional para construir los árboles B.

Se hicieron algunas adiciones específicamente para el SOB, puesto que ciertos métodos no estaban incluidos en la versión disponible. En particular, se programaron e implementaron

- actualización de listas de marcas de un par (contexto, valor)
- operación “union-all”
- eliminación de instancias de una lista de resultados con repetición (podía haber instancias repetidas) que no tuvieran por lo menos N repeticiones.

Sin embargo, en los últimos dos años se elaboró una versión nueva del KBC (Hernández, 2010) que construía y usaba los árboles con programas específicos, de modo que ya no usa bases de datos, sino sólo archivos planos.

Estas circunstancias hicieron que se realizaran los procesos tipo ETL (la carga de datos a partir de los de la base de datos relacional) en ambas versiones. Para las consultas sólo se usó la primera versión, la que reemplazó los árboles por los índices de una base de datos. El resultado fue que los datos que se obtuvieron fueron:

- espacio en disco utilizado por el SOB en DBB: de la versión nueva
- tiempos de respuesta de consultas: de la versión anterior.

No se repitieron las consultas usando la nueva versión del KBC, puesto que los tiempos de respuesta serían menores a los anteriores, pero la diferencia sería de milisegundos. Esta afirmación se hace tras probar algunas consultas y determinar sus tiempos de respuesta en la nueva versión, y compararlos con los equivalentes obtenidos originalmente.

8.6 Carga de datos simulados (ETL) a la estructura definida

Introducción

A continuación se efectúan los procesos correspondientes a lo que se denomina ETL (external table loading): se usan los datos (simulados) generados en la base de datos para crear las UBI's en la estructura DBB, de acuerdo a las clases y sus atributos definidos en la etapa anterior.

Se decidió utilizar los datos ya generados, en lugar de generarlos nuevamente, puesto que este método permitiría determinar las duraciones de procesos típicos de actualización de datos de una bodega de datos a partir de datos de otras fuentes, en particular, bases de datos.

Este proceso se realizó en etapas, que se presentan una vez más, como sub-secciones de esta seccion.

1. Diseño del proceso ETL
2. Diseño, programación y pruebas de los programas
3. Creación de fichas de libros
4. Creación de fichas de libros
5. Actualización del contexto 12 con los clientes que compraron cada libro
6. Actualización de los vectores paralelos de clientes
7. Analizar los tiempos de carga de datos
8. Modificar los programas de carga si fuera necesario

8.6.1 Diseño del proceso ETL

Se determinó que el proceso se dividiría en 4 etapas principales, mismas que se podrían ejecutar en forma parcial, si las duraciones fueran excesivas, donde este término refleja que se violarían ciertas restricciones que pudieran resultar de los equipos de cómputo y condiciones de operación de los mismos. Por ejemplo, una proceso que dura varios días sólo se ejecutaría si fuera posible reiniciarlo si hubiera alguna interrupción. Otra restricción podría ser la capacidad de algún dispositivo, o la cantidad de memoria disponible para un proceso.

Se adoptó un criterio que se incorporó al diseño de los procesos de carga de datos: los procesos deberían ser divisible, es decir, se podrían ejecutar en varias

ejecuciones, y que si fallara algún proceso, no sería necesario repetir todos los anteriores.

Las 4 etapas del proceso ETL son las siguientes:

1. Creación de las fichas de libros, con sus atributos. Se marcarían los campos: tipo de libro, tema, año de publicación, autores, palabras clave, idioma original, idiomas adicionales.)

2. Creación de las fichas de clientes, con sus atributos. se marcarían los campos categoría del cliente, país, ciudad.

3. Actualización de los clientes que compraron un libro. Se crearían las marcas del contexto 12 (clientes que compraron el libro) para cada libro, y las instancias serían los números de las fichas de los clientes correspondiente.

4. Creación de los vectores paralelos de las fichas de clientes: libros que compraron, y para cada libro, el año y mes de la operación.

Se resumen los programas, pero se los incluyó en código fuente en el CD anexo de esta tesis. Es importante señalar que para efectos de los resúmenes se han cambiado algunos nombres, especialmente de clases y variables.

8.6.2 Diseño, programación y pruebas de los programas

Se diseñaron los programas para cada una de estos 4 procesos, pero se elaboró un único programa que realizaría todas ellas. Como parte del diseño de las funciones y algoritmos, se impuso la condición de que sería relativamente sencillo hacer cambios si surgiera la necesidad de hacerlos, especialmente derivados de duraciones excesivas de los tiempos de ejecución. De hecho sucedió precisamente esa circunstancia. La versión descrita de los programas es la definitiva, a pesar de que hubo varias versiones anteriores.

En lugar de describir primero el diseño de los programas, se incorporó éste a la descripción de los procesos correspondientes.

Observación: como los programas se elaboraron en Visual Basic 6.0, se utilizó la terminología de este lenguaje de programación, en particular, el término “recordset”.

8.6.3 Creación de fichas de libros

Decisiones

- El proceso se haría por tipo de libro (a pesar de que el proceso no sería muy tardado, se prefirió hacerlo de este modo.) Esto demoraría el “marcado” de las ficha creadas, pero ofrecería la posibilidad de repetir procesos sin tener que hacer lo propio con los que lo precedieron.
- El número de UBI (ficha) es el mismo que el número de libro en la base de datos.

El programa mismo

Preparación

1) Conectar la base de datos del SOB.
2) Determinar el tipo de libro a procesar (el usuario lo indica en una forma.)
3) Preparación de los recordsets que usará el programa (los datos fuente)
    - Crear un recordset con los libros de ese tipo, ordenados por número de libro, a partir de la tabla “LIBROS” de la base del datos.
    - Crear un recordset con los autores de los libros del tipo seleccionado, ordenados por libro, a partir de la tabla “Libro-autor” de la base de datos.
    - Crear un recordset con las palabras clave de los libros del tipo seleccionado, ordenados por libro, a partir de la tabla “Libro-palabraclave” de la base de datos.
    - Crear un recordset con los idiomas adicionales de los libros del tipo seleccionado, ordenados por libro, a partir de la tabla “Libro-idioma” de la base de datos.
4) Preparar los archivos del DBB para las fichas.
5) Crear una instancia del KBC, y pasarle los contextos que se usan.
6) Para cada libro del recordset
    - crear la ficha en memoria, con los campos fijos indicados para esta clase
    - marcar los campos marcables (invocando el KBC)

- actualizar los idiomas adicionales del libro: se agregan como marcas individuales (o aisladas) y marcar estos idiomas
- actualizar las palabras clave del libro: se agregan como marcas individuales (o aisladas) y se marcan estas palabras clave
- actualizar los autores del libro: se incluyen en el vector de autores y se marcan las dos primeras palabras del nombre de cada autor con los contextos 701 y 702 respectivamente.
- grabar el arreglo de marcas aisladas
- armar y grabar el texto no marcable
- Tras actualizar los apuntadores indicados (a la lista de marcas aisladas, al texto no marcable y al vector de dimensión variable de autores) se graba la ficha.

Comentarios técnicos

Como el proceso descrito era muy tardado, puesto que enviaba las marcas al KBC “una por una”, se descartó el módulo que efectuaba el marcado y se lo reemplazó por otro.

Para cada contexto involucrado, se crea una instancia de una clase (valores-de-un-contexto)

- a medida que aparecen, se incluyen los valores del contexto invocados por una marca en un arreglo (ordenado por valor).
  - para cada uno de los valores se crea una instancia de la clase instancias-de-un-valor.

Cuando se invoca el método para agregar una marca (contexto, valor, instancia) el programa agrega ese número de instancia a la lista del valor.
Inicialmente se enviaban los datos al KBC cada vez que una de estas listas pasaba de cierto número de elementos, pero en la versión final se eliminó esta acción. Al final del proceso de un tipo de libro, se enviaron todas las listas al KBC para su inclusión como marcas.

Cabe señalar que la primera modificación (la que sustituyó el marcado uno por uno) redujo los tiempos de ejecución en factores del orden de 50 en algunos

de los procesos. La segunda modificación produjo otra reducción muy significativa. Los tiempos de estas cargas reflejan los de la versión final de la creación de fichas de libros.

8.6.4 Creación de fichas de clientes

Decisiones

- El proceso se haría por categoría de clientes. Puesto que se trataba de crear 5 millones de fichas, la probabilidad de una ejecución ininterrumpida era pequeña.
    - El proceso por categoría demora el marcado de las ficha creadas, pero ofrece la posibilidad de repetir procesos sin tener que hacer lo propio con los que lo precedieron.
- El número de UBI (ficha) es el número de cliente en la base de datos + 10,000,000 (es decir, se le sumaron diez millones al número de cliente.) Esto se debe a que en la base de datos (fuente de estos procesos) los libros están numerados del 1 en adelante, y lo mismo sucede con los clientes. Como las UBI's tienen un número único, había que resolver el conflicto entre una ficha correspondiente al libro 1 y otra ficha correspondiente al cliente 1.

El programa mismo

Preparación

- Conectar la base de datos del SOB.
- Determinar la categoría de clientes a procesar (el usuario la indica en una forma.)
- Crear un recordset con los clientes del esa categoría, ordenados por número de cliente, a partir de la tabla "CLIENTES" de la base del datos.
- Preparar los archivos del DBB para las fichas.
- Crear una instancia del KBC, y pasarle los contextos que se usan.

Para cada cliente del recordset

- crear la ficha en memoria, con los campos fijos indicados por la clase

- marcar los campos marcables (invocando el KBC)
- Armar y grabar el texto no marcable
- Armar la lista de marcas aisladas y grabarla
- Tras actualizar los apuntadores indicados (a la lista de marcas aisladas y al texto no marcable) se graba la ficha.

Comentarios técnicos

Como el proceso descrito era muy tardado, puesto que enviaba las marcas al KBC “una por una”, se descartó el módulo que efectuaba el marcado y se lo reemplazó por otro.

Para cada contexto involucrado, se crea una instancia de una clase (valores-de-un-contexto)

- a medida que aparecen, se incluyen los valores del contexto invocados por una marca en un arreglo (ordenado por valor).
    - para cada uno de los valores se crea una instancia de la clase instancias-de-un-valor.

Los comentarios al proceso de la creación de las fichas de libros aplican – con las modificaciones pertinentes – a la creación de las fichas de clientes, excepto que en este caso, los procesos eran tan tardados que resultaban imprescindibles dichas modificaciones.

8.6.5 Actualización del contexto 12 con los clientes que compraron cada libro

Para efectuar este proceso, se tomaron dos decisiones. La primera fue separar este proceso del que se describe en la sección siguiente (la creación de los vectores de libros comprados por cada cliente.) Inicialmente se pensó en ejecutar estos procesos en forma conjunta: crear el vector de libros comprados, y marcar cada uno con el contexto correspondiente (12). Esto evidentemente presentaba un aspecto desfavorable.

Para crear estos vectores, se procesan las ventas (de la tabla Libro-cliente) en orden de los clientes (y número de libro). Esto resultaría que, para cada libro que

se incluyera en el vector, se crearía una marca cuyo valor es el número de libro. Aunque se podría haber hecho esto, hubiera resultado en la creación de 300.000 instancias de una clase, circunstancia que hubiera puesto exigencias enormes de memoria disponible.) Al dividir el proceso, para el marcado se ejecutaría el proceso en orden “número de libro - número de cliente”, de modo que con una única instancia – el número de libro – se armaría el arreglo de clientes, y tras invocar el marcado correspondiente se destruye la instancia.

La otra decisión fue que el proceso se haría por tipo de libro, para evitar corridas de duración excesiva. De hecho, tras aplicar todas las modificaciones y mejoras a los programas de carga de datos, se comprobó que se pudo haber hecho la carga en una ejecución única, como se desprende del tiempo total que demoró este proceso (obtenido sumando las duraciones del proceso para cada uno de los 6 tipos de libro).

Preparación: conectar la base de datos del SOB

Crear un recordset con las ventas (libro, cliente; la fecha de compra no interviene en este proceso) ordenados por **número de libro y número de cliente**, a partir de la tabla “Libro_cliente” de la base del datos.

> *Observación: este proceso no graba datos del DBB; sólo genera marcas en KBC.*

Crear una instancia del KBC, y pasarle el contexto 12 (el único que se usa en este proceso).

El programa usa una lista única de enteros de 4 bytes, en los que incluye los clientes que compraron un libro.

Para cada registro del recordset

> Se “corta” por libro, es decir, se procesan los registros correspondientes a cada libro
>
> Para cada registro (del recordset) del libro, se agrega el número de cliente (+ 10,000,000) a la lista de clientes.

Cuando termina un libro (es decir, comienzan las ventas del siguiente o cuando se acaban los registros del recordset), se actualizan las marcas de ese libro (la lista de clientes se envía con contexto 12 al KBC) y se “prepara la siguiente lista” (vacía.)

Comentarios técnicos

La versión inicial (enviando cada cliente de un libro como marca al KBC) mostró duraciones intolerables. Puesto que en total se procesarían 203 millones de registros, la actualización de las marcas resultaba en tiempos de muchas horas aun cuando se efectuaban con lotes pequeños de libros.
Cuando se enviaron las marcas como listas, estos tiempos se redujeron de modo que la carga total duró 2 horas.

8.6.6 Actualización de los vectores paralelos de clientes

Aquí también se tomó una decisión parecida a la de la etapa anterior de la carga tipo ETL; el proceso se haría por categoría de clientes, o aún en lotes menores si fuera preciso.

Preparación: conectar la base de datos del SOB
Crear un recordset con las ventas (libro, cliente, la fecha de compra) ordenados por **número de cliente y número de libro**, a partir de la tabla “Libro_cliente” de la base del datos.

Observación: este proceso no genera marcas, sólo datos del DBB.

El programa usa una lista única de enteros de 4 bytes, en los que incluye los libros que compró un cliente.
Para cada registro del recordset

se “corta” por cliente, es decir, se procesan los registros correspondientes a cada cliente.

Para cada registro del cliente, se agrega el número de libro a la lista de libros.

Cuando termina un cliente, se graba la lista como vector, y se actualizan, en la ficha del cliente, tanto la dimensión de la lista como el apuntador a su ubicación en el archivo correspondiente y se “prepara la siguiente lista” (vacía.)

Comentarios técnicos

Como este programa no usa el KBC, la versión inicial fue adecuada, excepto en el modo en el que grababa los vectores resultantes. El estudio de este proceso resultó en un cambio en el modo en el que el DBB almacena los vectores de dimensión variable.

8.6.7 Analizar los tiempos de carga de datos

Como uno de los aspectos de interés era determinar las duraciones de procesos típicos de ETL usando el DBB, se registraron los tiempos de carga a estas estructuras. Como se mencionó, puesto que se obtuvieron tiempos muy prolongados, se rediseñaron los procesos y, tras adecuar los programas correspondientes, se repitieron las cargas de datos. tal cual se anunció, no se detallan en esta tesis los pasos intermedios (en la sección anterior se describió el proceso en la última versión de los mismos.)

8.6.8 Modificar los programas de carga si fuera necesario

Estas modificaciones se describieron en las secciones anteriores. La presencia de esta sección refleja la planificación de esta parte del proyecto, misma que se cumplió en las etapas y actividades indicadas en el plan del proyecto. En especial cabe señalar que, al haber planeado los programas de modo que ciertas modificaciones serían fáciles de programar y probar, el trabajo causado por estos cambios no resultó en cargas adicionales de programación.

CAPÍTULO 9. LAS CONSULTAS CON DBB

9.1 La consulta básica

Las consultas en DBB están basadas en las palabras clave por contexto. Se formulan consultas individuales a KBC, que tienen la forma: arma la lista de todas las instancias del par (contexto = c; valor = v). En la versión utilizada en esta investigación, KBC regresa esta lista como un archivo en disco.

KBC también ejecuta fórmulas, que son operaciones booleanas con operandos de dos tipos: listas de resultados armadas previamente o pares (contexto, valor.) Las operaciones que efectúa con la Unión, la Intersección, la diferencia y la "Unión ALL". En esta última, se arma una lista con todos los elementos que están en las listas indicadas por los operandos. La diferencia con la Unión estriba en que un elemento que aparece en más de una lista estará repetido en la lista resultante.

Como se ejecutarían muchas miles de consultas, se hizo un programa que no sólo invocaría una tras otra varias consultas, sino que registraría en una base de datos los tiempos de respuesta obtenidos para cada una de ellas.

Recordamos que la consulta básica del SOB es:

Cuando el cliente **C1** adquiere el libro **L1**,

- Armar la lista de todos los libros
  - que compraron los clientes que compraron **L1**.

De esta lista se eliminan los libros que ya compro el cliente **C1**.

En DBB, esta consulta se hace en pasos.

Paso 1: Armar la lista RL-CL1 de las instancias del par (contexto = 12, valor = L1) donde el contexto 12 es "clientes que adquirieron un libro."

Paso 2. Optativa (si se incluyó esta condición): eliminar de esta lista los clientes de la categoría 6. Para ello, se obtiene RL-CL2 con el par (contexto = 2, valor = 6) donde el contexto 2 es la categoría del cliente. A continuación se obtiene RL-CL3 = (RL-CL1 and NOT RLCL3), es decir, la diferencia entre las dos listas.

Si no se desea imponer esta condición, se renombra RL-CL1 como RL-CL3.

Paso 3. Construir la lista RL-LIB1 de libros como unión de todos los valores de los vectores “1” de cada cliente de la lista RL-CL3). El vector 1 de la clase clientes contiene los libros que adquirió cada cliente.

Paso 4: Construir la lista RL-LIB2 de los libros que compró el cliente C1 (del vector 1.)

Paso 5: Obtener la lista “resultado” como diferencia entre RL-LIB1 y RL-LIB2 (RL_LIB1 and NOT RL_LIB2.)

Observación técnica

En el proceso efectuado, en lugar de usar los métodos del KBC para hacer las operaciones entre listas de libros, se programaron rutinas para hacerlo. Se usó un arreglo de bytes, cada uno de los cuales representa un libro, y el valor 1 indicaba que dicho libro había sido adquirido por algún cliente. Para el paso 5, se “apagaron” los bytes correspondientes a libros que había comprado el cliente C1. Observe que, si el número de libros hubiera sido mucho mayor (en este caso, fueron 300.000 libros) consideraciones de memoria podrían haber motivado el uso de cadenas de bits en lugar del arreglo de bytes. De hecho, eso es precisamente lo que hace el KBC cuando debe efectuar operaciones entre conjuntos muy numerosos.

Ilustraremos el proceso implementado con un ejemplo sencillo mediante la Tabla 31. Supongamos que hubiera 18 libros (numerados del 1 al 18), y que el cliente 1 compró los libros 2,4,8 y 11, el cliente 2 los libros 3, 8, y 14 y el cliente 3 los libros 8, 14, 15 y 16), donde el libro que dio lugar a la consulta (B1) es el número 8. Supongamos que el cliente C1 había adquirido previamente los libros 7 y 15, además del 8 que acaba de comprar.

Tabla 30. Ilustración del proceso de unión de libros adquiridos por los clientes y la eliminación de los que ya había comprado el cliente C1

Después de	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
INICIO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cliente 1		**X**		**X**				**X**			X							
	0	**1**	0	**1**	0	0	0	**1**	0	0	1	0	0	0	0	0	0	0
Cliente 2			**X**					**X**						X				
	0	**1**	**1**	**1**	0	0	0	**1**	0	0	1	0	0	1	0	0	0	0
Cliente 3								**X**						X	**X**	**X**		
	0	**1**	**1**	**1**	0	0	0	**1**	0	0	1	0	0	1	**1**	**1**	0	0
Cliente C1							**X**	**X**							**15**			
Lista final	0	**1**	**1**	**1**	0	0	**0**	**0**	0	0	1	0	0	1	**0**	**1**	0	0

Observación: no importa si en el proceso de unir las listas de libros se tomó en cuenta o no el cliente C1 (de hecho, el programa no lo hace.)

9.2 Duraciones obtenidas al efectuar las consultas en DBB

Se ejecutaron los mismos lotes de consulta que se reflejan en los cuadros finales usando la BDR (con SQL.) Además, se ejecutaron los lotes 3 veces cada uno, para determinar la variación en las duraciones, mismas que se muestran en la Tabla 32.

Tabla 31. Duraciones en segundos y fracción de la consulta básica sin excluir los clientes de la categoría 6. 3 ejecuciones de cada lote.

L1: Libro #	313800	317201	318621	320781	320780
Ventas	**1064**	**5001**	**10601**	**20001**	**39951**
C1: Cliente	1542576	4933550	1861521	4562410	3212978
Cuántos libros compro C1	10	386	7	66	43
Total libros en lista final	4547	11490	22693	39078	73804
Corrida 1	0,20	0,63	1,31	2,49	5,45
Corrida 2	0,27	0,78	1,66	3,17	6,19
Corrida 3	0,22	0,64	1,30	2,48	4,92
Promedio	**0,23**	**0,68**	**1,42**	**2,71**	**5,52**

Como se ve, la variación es mínima, de modo que en los restantes cuadros sólo se reflejó el promedio de las duraciones.

Resumen de las duraciones de las consultas efectuadas con el DW en DBB

Las tablas 33 y 34 muestran estas duraciones, para las consultas incluyendo y excluyendo a los clientes de la categoría 6, es decir, aquéllos que adquieren "muchos" libros.

Tabla 32. Resumen de consultas incluyendo los clientes de categoría 6 (en segundos con 2 decimales)

L1: Libro #	313800	317201	318621	320781	320780
Ventas	**1064**	**5001**	**10601**	**20001**	**39951**
Total libros en lista final	4547	11490	22693	39078	73804
Duración	**0,23**	**0,68**	**1,42**	**2,71**	**5,52**

Tabla 33. Resumen de consultas excluyendo los clientes de categoría 6 (en segundos con 2 decimales)

L1: Libro #	313800	317201	318621	320781	320780
Ventas	**1064**	**5001**	**10601**	**20001**	**39951**
Total libros en lista final	1608	5501	11730	19608	39673
Duración	**0,30**	**0,62**	**1,16**	**2,04**	**3,85**

9.3 Consultas con criterios de selección adicionales

En situa0ciones de análisis de negocios, se pueden incluir otro tipo de restricciones a la lista de libros que se ofrecerán al cliente. Se ha dedicado un capitulo especial a este tipo de consultas (el capítulo 12.)

CAPÍTULO 10. COMPARACIONES ENTRE LAS 2 TECNOLOGÍAS

Introducción

De acuerdo al objetivo del proyecto, se compararían dos aspectos fundamentales de la implementación en las dos tecnologías de la situación de negocios.

El espacio en disco total necesario para almacenar todos los datos del DW.

Los tiempos de respuesta obtenidos para las consultas en ambas tecnologías.

Espacio en disco

El espacio (en Gigabytes) que ocupó el acervo de datos en las diversas bases usando SQL-SERVER (dependiendo de los índices de la tabla de VENTAS) se muestra para las 7 bases se muestra en la Tabla 35.

Tabla 34. Espacio en disco ocupado por el acervo de datos en las 7 bases de datos relacionales

BASE #	Libro y cliente	Cliente y Libro	Cliente	Libro	Espacio en disco
1	P-CL		S		9.34
2	P-CL				7.63
3	S		P-CL		12.7
4	P		S		14.1
5		P-CL			7.63
6			P-CL		9.13
7		P-CL		S	9.34
DBB					

P = índice principal, **P-CL** = índice principal clustered, **S** = índice secundario

Observe que se agregó el espacio ocupado por el acervo usando las estructuras del DBB.

Aquí cabe analizar la causa de las variaciones, pero también el impacto que tienen los índices.

- Observe que la ausencia de un índice naturalmente reduce el espacio en disco. Pero la duración de las consultas se incrementa en forma notable, como se vio en el Capítulo 7.

- El uso de un índice cluster, como no crea una estructura sino ordena los registros de acuerdo a los valores del índice, también tiene un efecto significativo sobre el tamaño del acervo de datos. Además, la presencia de un índice agrupado, como vimos, reduce las duraciones de las consultas.

Para obtener alguna conclusión, se comparan el espacio usado por DBB con el de las bases B1 y B7, que son los mismos. Los tamaños se muestran en la tabla 36.

Tabla 35. Espacio en disco ocupado en ambas tecnologías (con la Base #1)

SQL	DBB	FACTOR DBB / SQL	FACTOR SQL / DBB
9.34	**4.2**	**0.45**	**2.22**

Como se verá en el capítulo 11 (efecto de la duplicación del tamaño del acervo de datos) la relación se mantiene prácticamente igual, aunque en SQL el espacio es un poco menos que el doble del original. En DBB, es prácticamente lineal el incremento.

Comparación de las duraciones de las consultas en ambas tecnologías

En la tabla 37 se muestra un resumen de los tiempos de respuesta a la consulta básica (con y sin los clientes de categoría 6) en las 7 bases de datos y los correspondientes en DBB. Como en las consultas efectuadas en DBB no se incluyó el libro número 9521 (se usó otro libro de un número similar de ventas) se eliminó éste de las comparaciones. Los datos que refleja esta tabla corresponden a los cuadros resumen del capítulo 7 para los de SQL, mientras que los del DBB son los del correspondiente cuadro del capítulo 9. Recuerde que en ambos resúmenes se reflejaron solamente los tiempos promedio.

Tabla 36. Comparación de tiempos de respuesta a la consulta básica en las 7 bases de datos y en DBB

L1: Libro #		313800	317201	318621	320781	320780
Ventas		1064	5001	10601	20001	39951
Con clientes de categoría 6						
Total libros en lista final		4547	11490	22693	39078	73804
En **SQL**	Base #1	0,90	0,91	1,82	3,82	7,89
	Base #2	465,25	449,13	435,12	433,02	432,45
	Base #3	2,00	4,76	12,84	24,67	23,83
	Base #4	1.13	1.04	1.92	4.76	17.76
	Base #5	383,08	374,85	370,34	370,80	384,31
	Base #6	745.20	787.63	760.16	757.20	843.53
	Base #7	0,51	1,20	2,43	7,45	19,98
en **DBB**		**0,23**	**0,68**	**1,42**	**2,71**	**5,52**
Excluyendo a los clientes de categoría 6						
Total libros en lista final		1608	5501	11730	19608	39673
En **SQL**	Base #1	1,07	0,75	1,51	3,17	6,05
	Base #2	449,99	443,09	450,59	470,26	534,20
	Base #3	5,22	3,56	8,33	16,45	28,22
	Base #4	1.04	0.76	1.75	3.78	15.37
	Base #5	382,53	373,13	376,14	383,84	478,01
	Base #6	No procesada				
	Base #7	1,58	1,08	2,32	6,19	11,45
en **DBB**		0,30	0,62	1,16	2,04	3,85

En lugar de presentar los impactos numéricos (reducción del tiempo de respuesta resultante de usar las estructuras del DBB en lugar de una base de datos relacional) sólo se calcularon los “factores de incremento” y los porcentajes de reducción comparando los obtenidos usando la mejor base de datos (en cuanto a tiempos de respuesta, es decir, la Base #1.

En la Tabla 38, el factor SQL/SOB indica que la consulta en SQL demoró (ese factor) veces más que en DBB. Por ejemplo, 0.91 es 1.34 veces mayor que 0.68. La reducción porcentual se refiere a la disminución de la duración en SQL. Por ejemplo, 0.91 – 0.68 = 0.23 que es el 25% de 0.91 (los porcentajes se redondearon a enteros.)

Tabla 37. Comparación de tiempos de respuesta a la consulta básica usando la mejor base (Base #1)

L1: Libro #	313800	317201	318621	320781	320780
Ventas	1064	5001	10601	20001	39951
Total libros en lista final	4547	11490	22693	39078	73804
Con clientes de categoría 6					
Duración en SQL	0.90	0.91	1.82	3.82	7.89
en DBB	0.23	0.68	1.42	2.71	5.52
Factor (DBB / SQL)	3.91	1.34	1.28	1.4	1.42
Reducción porcentual	74 %	25 %	22 %	29 %	30 %
Excluyendo a los clientes de categoría 6					
Duración en SQL	1.07	0.75	1.51	3.17	6.05
en DBB	0.30	0.62	1.16	2.04	3.85
Factor (DBB / SQL)	3.56	1.21	1.30	1.55	1.57
Reducción porcentual	72 %	17 %	23 %	35 %	36 %

Conclusiones de estas comparaciones

Como se ve, el impacto de los tiempos promedio no es espectacular, aunque en algunos casos sí se reducen significativamente las duraciones. Pero hay que tomar en cuenta que se tomó una base de datos con los índices necesarios para las consultas que se probaron, y se usó el índice principal agrupado, circunstancia que reduce notablemente los tiempos de respuesta. Además, se usaron los tiempos "promedio", que como se vio, eliminan las consultas muy prolongadas puesto que se las promedia con otras, ya sea repetidas o ejecutadas en orden diferente.

La ausencia de un índice, que es la situación que se presenta en análisis de negocios puesto que frecuentemente se hacen consultas no previstas, ocasiona consultas con tiempos de respuesta muy prolongados. En DBB no sucede esto, porque si bien no usa índices, sí tiene los contextos necesarios definidos.

Una de las conclusiones se refiere al espacio en disco. La inclusión de un índice en una tabla de una base de datos relacional resulta en un considerable aumento del espacio en disco que ocupa, mientras que en DBB, los contextos – aunque naturalmente ocupan espacio - éste es considerablemente menor que un índice en una base de datos.

CAPÍTULO 11. EFECTO DE LA DUPLICACIÓN DEL TAMAÑO DEL ACERVO

Introducción

Con el objeto de determinar ciertos aspectos en cuanto a escalabilidad en DBB, se repitieron todos los procesos pero para el doble de número de libros, número de clientes y número de ventas. Esto incluyó la generación de datos simulados, la ejecución de las consultas con SQL, el proceso ETL de implementación del SOB en DBB y las consultas en esta tecnología.

Es importante señalar que lo que se hizo no es suficiente para determinar lo que sucedería. Esto se debe a que, aunque se duplicaron los datos, no se modificaron las ventas de un libro o las compras de un cliente, comparadas con las de la base original (es decir, antes de duplicar) sino que se agregaron libros, clientes y ventas para estos nuevos libros y clientes.

Como se mencionó en el Capítulo 2, la escalabilidad en general se debe a factores como la inclusión de “más datos” pero de los mismos conceptos. Por ejemplo, en un almacén de datos de ventas de una cadena de supermercados, se habían incluido ventas de 6 meses, y ahora se pretende incluir las de todo el año.

Sólo se presentan algunos datos representativos del impacto de la duplicación, aunque se hicieron muchas otras comparaciones, que arrojaron resultados similares en todas las versiones de la base de datos en SQL Server, y con todo tipo de lotes de consultas.

Para determinar el impacto del crecimiento de datos, se duplicó el acervo descrito. Se generaron 644.000 libros, 10.000.000 de clientes y 406 millones de ventas. Cabe señalar que no se generaron más ventas de los libros, sino simplemente se agregaron nuevos libros. Tras analizar los espacios en disco, se determinó que en ambas tecnologías, el incremento era casi lineal.

Espacio total en disco para el almacén de datos de ambos tamaños del acervo, mismos que se designan como el "original" y el "doble"

En la tabla 39 se muestran los datos obtenidos para la "mejor" base de datos, la llamada Base # 1, con el índice principal clustered: libro-cliente y un índice secundario:cliente.

Tabla 39. Espacio en disco ocupado por los acervos original y doble

	Original	**Doble**	**Factor crecimiento**
En SQL Server	**9.34**	**19.6**	**2.1**
En DBB	**4.2**	**8.4**	**2**

Tiempos de respuesta a consultas

Cabe señalar que se probaron muchos lotes, es decir, con diferentes libros. Sin embargo, la relación entre los tiempos era aproximadamente la misma, de modo que se decidió adoptar el lote que se describe más abajo para documentar los tiempos de respuesta.

Se muestran en la Tabla 40 los resultados para un lote "típico" ejecutado en la Base #1 y en DBB con **el doble de libros, clientes y ventas**. Todas las duraciones son promedio, en el sentido explicado en la sección de consultas en SQL. El factor SQL/SOB índice que la consulta en SQL demoró (ese factor) veces más que en DBB. Por ejemplo, 5.18 es 11.25 veces mayor que 0.46.

La reducción porcentual se refiere a la disminución de la duración en SQL. Por ejemplo. 0.88 – 0.16 = 1.72 que es el 81.80% de 0.88 (los porcentajes se redondearon a enteros.)

Tabla 38. Comparación de tiempos de respuesta para el acervo doble

Libro # (L1)	109492	610366	634401	631421	641563	640780
Ventas	20	1064	5001	10601	20001	39951
SQL BASE 1	0.88	1.05	2.17	7.93	7.24	7.38
DBB	0.16	0.46	1.17	1.94	3.47	7.09
Factor SQL/SOB	**18**	**44**	**54**	**24**	**48**	**85**
Reducción porcentual	**82**	**56**	**46**	**76**	**52**	**15**

Observación: por un error en el registro de las consultas, no se incluyó el número de libros de la lista final en este cuadro. Sin embargo, ese dato carece de importancia en cuanto a la comparación, por dos motivos. Primero, naturalmente es el mismo en ambas tecnologías (es la misma consulta) y durante todos los procesos, siempre se verificó que éste era el caso. Pero además, el número de libros resultantes era similar al del acervo inicial, el de 5,000.000 de clientes, puesto que como se explicó, sólo se agregaron más libros y más clientes, pero no se aumentaron las ventas de algún libro ni las compras de alguno de los clientes originales.

Conclusiones de estas comparaciones

El impacto de la duplicación es similar en ambas tecnologías. exceptuando las “primeras” consultas en dos sentidos:

- la primera de cada lote de consultas
- la primera de esa sesión de trabajo.

En DBB los tiempos de respuesta son casi idénticos (difieren a los sumo en unos milisegundos.) Este resultado era predecible puesto que el tamaño no afecta las consultas, excepto quizá en el número de archivos que se abren para leer los datos de los árboles y de las fichas.

En cambio en SQL. el resultado nos sorprendió. Fue uno de los motivos para estudiar profundamente como funcionaba la versión del SQL Server que utilizamos. Ahí encontramos la explicación de esta circunstancia (tiempos de respuesta casi iguales a las de la base original.) El manejador arma en memoria todo lo que obtiene en cada paso de sus ejecuciones de consultas. de modo que muchas actividades se hacen en memoria.

De hecho, cuando ejecutamos las consultas en la base “doble” por vez primera, habíamos asignado 400 Gb de memoria para uso del SQL Server. Los tiempos de respuesta eran considerablemente mayores a los de la base original, especialmente cuando se ejecutaban distintas consultas en una misma sesión de trabajo. Cuando le asignamos 600 Gb de memoria estas diferencias casi desaparecieron.

CAPÍTULO 12. OTRAS CONSULTAS CONTEMPLADAS EN SOB

12.1 Introducción

Se han mencionado otras consultas además de la que se denominó básica. En particular. la inclusión de condiciones impuestas a los libros a ofrecer. a características del cliente o a la fecha de adquisición. tanto del libro en cuestión como de otros libros. produce cambios (o adiciones de condiciones) en las sentencias SQL o en el modo en el que se efectúa la consulta en DBB. En esta sección se describen algunas de estas diferencias y su impacto sobre las duraciones.

12.2 Las condiciones que se probaron

Se describen las condiciones impuestas a los libros a ofrecer (el resultado de la consulta.) En cada caso. se entiende la condición como

no ofrecer libros que no cumplan con la condición especificada.

Se usa la notación habitual: la consulta se formula para el libro L1 y el cliente C1 que lo adquiere. El número de referencia corresponde al número de sub-sección en el que se describe esa consulta. Se muestran en la Tabla 41 las 7 consultas que se describirán en este capítulo.

Tabla 39. Las 7 consultas detalladas

#	**Variable**	**Condición**
1	Libros	De ese tema
2	Libros	Que tengan por lo menos N palabras clave en común con L1 (N de 1 a 5)
3	Libros	Libros que se ofrecen en alguno de los idiomas en los que se ofrece L1
4	Fecha de compra de L1	Libros que compraron clientes que compraron L1 en los últimos M meses
5	Fecha de compra de otros libros	Libros que compraron en los últimos M meses los clientes que compraron L1
6	Cliente	Clientes que compraron por lo menos otro libro del tema de L1
7	Cliente	Del mismo país (o ciudad)

12.2.1 Sólo ofrecer libros del mismo tema que L1

En SQL: se agrega una cláusula WHERE que se aplica a la lista de libros final. la ofrecida al comprador. Esta condición se puede imponer de dos modos:

- Determinar el tema del libro (con unan consulta al efecto) e introducir ese tema como parámetro de la consulta final.

```
SELECT * From libros
   WHERE tema=TEMA_L1 and num_lib_dbb IN(
   SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente IN{
     SELECT num_cliente From clilibro Where num_lib_dbb=L1}
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1)
```

- Incluir la determinación del tema del libro en la consulta misma.

```
SELECT * From libros
   WHERE tema=(Select tema from libros where num_lib_dbb=L1)
         and num_lib_dbb IN(
   SELECT DISTINCT num_lib_dbb From clilibro
      WHERE num_cliente IN{
     SELECT num_cliente From clilibro Where num_lib_dbb=L1}
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1)
```

En DBB

1. Se consigue el tema de L1 (de la ficha del libro)
2. Se interseca la lista final con la lista de resultados obtenida con la consulta al par (contexto: 3. Valor: el tema de L1) puesto que el contexto 3 es el tema del libro.

Sólo ofrecer libros que tengan por lo menos N palabras clave en común con L1

Esta consulta se formula para cualquier N. entre 1 y 5, pero naturalmente N no puede ser mayor al número de palabras clave indicadas para el libro L1.

En SQL

```
SELECT * FROM LIBROS WHERE num_lib_dbb
 IN(
     SELECT num_lib_dbb FROM libroskeyword WHERE num_lib_dbb
     IN(
         SELECT num_lib_dbb From libros WHERE num_lib_dbb
         IN(
             SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
             IN(
                SELECT num_cliente From clilibro Where num_lib_dbb=L1
               )
             EXCEPT
             SELECT num_lib_dbb From clilibro Where num_cliente= C1
           )
       )
     AND keyword
     IN(
        Select keyword from libroskeyword where num_lib_dbb=L1
       )
     GROUP BY num_lib_dbb HAVING COUNT(*)>=N_PALABRAS_COMUN
   )
```

En DBB

1. Se hace la consulta “básica”. A continuación. se determinan los libros que cumplen la condición del siguiente modo.
2. e arman las listas de resultados para cada una de las palabras clave del libro L1. Es decir. para i = 1 a número de palabras clave indicadas para L1
3. LR (i) = contexto: 6. valor: palabra clave i del libro L1 (puesto que el contexto 6 es “palabra clave”.
4. Se hace una “Union all” de estas listas (es decir. los libros pueden estar repetidos en la lista resultante). Esto resulta en una lista de resultados que llamamos LUA.
5. Se crea una nueva lista de resultados LCC con los elementos de LUA que están por lo menos N veces en ella. (Hay un método en KBC que efectúa precisamente esta operación.)
6. Se interseca la lista resultante de la consulta básica con LCC.

12.2.3 Sólo libros que están disponibles en uno de los idiomas en los que se ofrece L1

En SQL

```
SELECT * FROM LIBROS WHERE num_lib_dbb
 IN(
   SELECT num_lib_dbb FROM librosidiomaadicional WHERE num_lib_dbb
   IN(
     SELECT num_lib_dbb From libros WHERE num_lib_dbb
     IN(
       SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
       IN(
          SELECT num_cliente From clilibro Where num_lib_dbb=L1
         )
         EXCEPT
         SELECT num_lib_dbb From clilibro Where num_cliente= C1
       )
     )
     AND idiomaadicional
     IN(
      Select idiomaadicional from librosidiomaadicional where
num_lib_dbb=L1
      union
      select idioma from libros where num_lib_dbb=L1
       )
     GROUP BY num_lib_dbb HAVING COUNT(*)>=1
union
     SELECT l1.num_lib_dbb From libros l1 WHERE l1.num_lib_dbb
     IN(
       SELECT DISTINCT cl.num_lib_dbb From clilibro CL WHERE
cl.num_cliente
       IN(
          SELECT cl1.num_cliente From clilibro cl1 Where
cl1.num_lib_dbb=L1
         )
         EXCEPT
         SELECT cl2.num_lib_dbb From clilibro cl2 Where
cl2.num_cliente=C1
       )
       and l1.idioma in(
 Select idiomaadicional from librosidiomaadicional where num_lib_dbb=L1
 union
 select idioma from libros where num_lib_dbb=L1
         )
)
```

En DBB

1. Encontrar los clientes que compraron el libro
2. Para cada uno de los clientes de la lista
   agregar los libros que compró ese cliente a la lista que se ofrecerá.
3. Se quitan los libros que ya compró el cliente C1.
4. Encontrar los idiomas en los que se ofrece L1 (de la ficha.)
5. Armar la lista LIDIORIG de libros ofrecidos en el idioma original de L1. con el par (contexto 7. valor; idioma original de L1). puesto que el contexto 7 es “idioma original del libro”.
6. Armar una lista de resultados LIDIADIC (i) para cada uno de los idiomas adicionales en los que se ofrece L1. con el par (Contexto: 8. valor = idioma adicional). puesto que el contexto 8 es el “idioma adicional”.
7. Armar la unión de las listas LIDIORIG y las LIDIACID (i).
8. Intersecar esta lista con la lista obtenida en el paso 3.

12.2.4 Sólo contemplar clientes que compraron L1 en los últimos M meses

Esta consulta se formula para cualquier M.

En ambos casos. se determina la “fecha de corte” a partir de la fecha del día. Para SQL. se usa la fecha (con el día). y para DBB. se usa el Año-Mes.

En SQL (la fecha de compra de un libro es un campo de la tabla Libro_cliente)

```
SELECT * From libros WHERE num_lib_dbb
IN(
   SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
   IN(
       SELECT num_cliente From clilibro Where num_lib_dbb=L1
              and fecha_compra>DATEADD(month. -M_MESES. getdate())
      )
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1
  )
```

Se incluye en la consulta la condición

clientes que compraron el libro

descartar los que lo compraron antes de la fecha de corte.

En DBB

La fecha de compra de un libro por un cliente está en un vector paralelo al de los libros que compró. De ese modo. la consulta se efectúa de este modo. donde los detalles de cada paso se mostraron anteriormente en el capitulo 9.

1. Encontrar los clientes que compraron el libro
2. Para cada uno de los clientes
   a. determinar si compró el libro después de la fecha de corte
   b. Si es el caso. agregar los libros que compró ese cliente a la lista que se ofrecerá
3. Se quitan los libros que ya compró el cliente C1.

12.2.5 Sólo se ofrecen libros que compraron en los últimos M meses los clientes que compraron L1

En ambos casos. se determina la fecha de corte como se describió más arriba.

En SQL se agrega al paso en el que se determinan los libros que compraron los clientes. la condición de que la fecha no sea anterior a la fecha de corte.

```
SELECT * From libros WHERE num_lib_dbb
IN(
   SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
   IN(
       SELECT num_cliente From clilibro Where num_lib_dbb=L1
      )
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1
          and fecha_compra>DATEADD(month. -M_MESES. getdate())
  )
```

En el DBB. la consulta se realiza del siguiente modo.

1. Encontrar los clientes que compraron el libro
2. Para cada uno de los clientes

   para cada libro que compró (del vector)

   agregarlo si el año-mes no es menor que la fecha de corte.
3. Se quitan los libros que ya compró el cliente C1.

12.2.6 Sólo incluir clientes que compraron por lo menos otro libro del mismo tema que L1.

En SQL.

```
SELECT * From libros WHERE num_lib_dbb
IN(
 SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
  IN(
     SELECT CL.num_cliente
     FROM clilibro CL INNER JOIN libros L
          ON CL.num_lib_dbb = L.num_lib_dbb
     WHERE tema=(select tema from libros where num_lib_dbb = L1)
           AND num_cliente = C1
     GROUP BY CL.num_cliente
     HAVING COUNT(*) >1
    )
 EXCEPT
 SELECT num_lib_dbb From clilibro Where num_cliente = C1
 )
```

En DBB. la consulta se realiza del siguiente modo.

1. Encontrar el tema del libro L1 (de la ficha del libro).
2. Armar la lista de resultados de los libros del mismo tema que L1. con el par (contexto 3. valor tema del libro L1).
3. Encontrar los clientes que compraron el libro (lista de resultados)
4. Para cada uno de los clientes

Obtener otra lista de resultados: Intersecar el vector de libros que compró el cliente con la lista de libros del mismo tema (sabemos que compró L1, que estará en la lista.)

5. si esta lista tiene por lo menos 2 elementos. agregar los libros del cliente a la lista a ofrecer.
6. Se quitan los libros que ya compró el cliente C1.

Observación: hay una alternativa. que consiste en quitar el libro L1 de la lista de libros del mismo tema que L1. y pedir que la lista del paso 4 no sea vacía.

12.2.7 Sólo tomar en cuenta clientes que residen en el mismo país o ciudad que C1 (el que dio lugar a la consulta.)

En SQL

```
SELECT * From libros WHERE num_lib_dbb
IN(
   SELECT DISTINCT num_lib_dbb From clilibro WHERE num_cliente
   IN(
     SELECT cl.num_cliente
      From clilibro cl. clientes c
      Where c.num_cliente = cl.num_cliente
      and num_lib_dbb=L1
      and (
          pais=(select pais from clientes where num_cliente=C1)
          or ciudad=(select ciudad from clientes where num_cliente=C1)
          )
      )
   EXCEPT
   SELECT num_lib_dbb From clilibro Where num_cliente = C1
  )
```

En DBB

1. Encontrar los clientes que compraron el libro
2. Encontrar los clientes que residen en ese país con el par (contexto 10. valor = país del cliente C1). Se haría de forma análoga con la ciudad. solo con el contexto 11 en lugar del 10.
3. Intersecar estas dos listas.
4. Para cada uno de los clientes de la lista
    a. agregar los libros que compró ese cliente a la lista que se ofrecerá.
5. Quitar de la lista los libros que ya compró el cliente C1.

CONCLUSIONES

Se implementó la situación de negocios denominada SOB en una base de datos relacional y en la tecnología DBB. Se generaron datos de prueba y se cargaron a ambas estructuras. Se hicieron las mismas consultas en ambas tecnologías y se compararon los tiempos de respuesta. Se repitieron las actividades que usaban una base de datos relacional en diversas bases que se distinguieron de las restantes por la presencia y naturaleza de los índices.

Las comparaciones permitieron determinar que el DBB resultaba en tiempos de respuesta más breves. pero especialmente eliminaba la gran variación en las duraciones de las consultas que se presentaron usando una BDR. A pesar de incluir el uso de índices agregados (clustered indexes) que redujeron notablemente el espacio en disco ocupado por las bases relacionales. el espacio total necesario para guardar los datos usando el DBB era mucho menor. aun incluyendo muchos campos marcados. que resultan en el equivalente a los índices de la base de datos relacional. Estos índices clustered mejoraron también los tiempos de respuesta de las consultas.

La conclusión a la que condujo la investigación descrita en esta tesis fue que el DBB ofrecía varias ventajas sobre las bases de datos relacionales. Como investigación futura se contempla repetir estas comparaciones con otras tecnologías. entre ellas las bases de datos columnares y otras basadas en el concepto de key/value. Como el objetivo del proyecto era determinar si valía la pena continuar el desarrollo del paquete DBB. se logró lo que se propuso. es decir. una respuesta a dicha interrogante. La respuesta es que vale la pena elaborar una versión del DBB. a pesar del enorme esfuerzo de desarrollo que ello significa.

REFERENCIAS

Abadi , Daniel J. 2008. Query Execution in Column-Oriented Database Systems. Massachusetts Institute of Technology.

Adabas Comprehensive Data Management System. 2010. ADABAS [Base de datos]. Disponible en: http://www.softwareag.com/corporate/products/adabas _ 2010/default.asp

Bain T. 2010. Data Analytics for the Masses [Base de datos]. Disponible en: http://www.readwriteweb.com/enterprise/2009/01/data-analytics-for-the-massesp2.php

Bartunov O. y Sigaev T. 2006. PostgreSQL Summit. Toronto. July 8-9 [Base de datos]. Disponible en: http://www.sigaev.ru/gin/Gin.pdf

Douglas K., Douglas S. 2003. PostgreSQL: a comprehensive guide to building, programming, and administering PostgreSQL Databases. Editorial Sams publisher. Primer edición. 791. p.p.

Bayer R. y McCreight E. 1972. "Organization and Maintenance of Large Ordered Indices". Acta Informatica Vol. 1 Fasc. 3. pp. 173-189.

Big Table. 2010. [Base de datos]. Disponible en: http://en.wikipedia.org/wiki/Big Table

Bloom, Burton H. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors Comunications of the ACM, Volumen 13 /Numero 7/ Pags 422-426.

Celkos, Joe. 2008 Thinking in Sets Auxiliary Temporal and Virtual Tables in SQL. Editorial Morgan Kaufmanm. Segunda edición. 362 p.p.

Codd E.F. 1970. A Relational Model of Data for Large Shared Data Banks. IBM Research Laboratory. San Jose. California. Communications of the ACM Volume 13 / Number 6 / June. pp 377-287.

Comer D. 1979. "The ubiquitious B-tree". Computing Surveys. VoL ll. No 2 11. pp. 121-137.

Conway D. 2010. Latest Teradata Database Release Supports Big Data and the Convergence of Advanced Analytics [Base de datos]. Disponible en:

http://www.teradata.com/t/News-Releases/2010/Latest-Teradata-Database-Release-Supports-Big-Data-and-the-Convergence-of-Advanced-Analytics/

Cruz Millán M. 2003. “Desarrollo de una interfase para establecer criterios para la recuperación de información a partir del uso de claves”. Tesis de maestría. Colegio de Postgraduados. México.

D L Childs. 2010. SETSTORE DATA ACCESS ARCHITECTURES “For High Performance Informationally Dense I/O Transfers”. XSP Technology Research & Development Series. Integrated Information Systems. Michigan
Disponible en: http://xsp.xegesis.org/SSDAA.pdf

Diaz S. 2010. Greenplum pushes enterprise data cloud with new releases [Base de datos]. Disponible en: http://www.zdnet.com/blog/btl/greeplum -pushes-enterprise-data-cloud-with-new-releases/32965

Elmasri and Navathe 1994. Sistemas de bases de datos conceptos funcamentales. Segunda edición, editorial Addison Wesley.

Frost S. 2010. DATA-Beat - A DATAllegro Blog [Base de datos]. Disponible en: http://www.beyeblogs.com/DATAllegro/archive/2008/
04/who_i_am_and_why_im_here.php

Gran Fritchey, 2008. Dissecting SQL Server Execution Plans. “the art of high-performance SQL code”, ed. Simple-Talk Publishing. 1 edition, 182 p.p.

Greenwald R.. Stackowiak R. y Stern J. 2004. Oracle Essentials: Oracle Database 11g. Boston. MA: O'Reilly.

Hernández Negrete. N. 2010. Un sistema de gestión de palabras claves por contexto para un acervo de datos. Tesis de Maestría. Colegio de Postgraduados. Disponible en.
http://www.biblio.colpos.mx:8080/jspui/bitstream/10521/244/1/Hernandez_Negrete_N_MC_Computo_Aplicado_2010.pdf

Hernández Negrete. N. Bauer-Mengelberg. J. 2010. Structures to store keywords by context. Enviado para publicación a IEEE. Transactions in Knowledge and Data Engineering.

Howard P. 2010. Netezza surprises with technical capabilities [Base de datos]. Disponible en: http://www.theregister.co.uk/2006/10/03/netezza_annual_ conference_roundup/

IBM. 2010. Data Warehouse InfoSphere Warehouse: Insight without Boundaries [Base de datos]. Disponible en: http://www.ibm.com/us/en/

Infobright open source data warehousing. 2010. Infobright Enterprise Edition [Base de datos]. Disponible en: http://www.infobright.com/Products/Features/

Inmon W.H. 2002. Building the Data Warehouse. NY: Wiley Computer Publishing.

Kallman Robert, et al. 2008 HStore "A HighPerformance, Distributed Main Memory Transaction Processing System". ACM.

Langgit Lynn. 2007. Foundations of SQL Server 2005 Business Intelligence.

Lugo Espinosa, Oziel. 2006. Formulación de consultas en las aplicaciones del estructurador de datos DBB. Colegio de postgraduados.

Madden Samuel. Et. al. 2010. Low Overhead concurrency Control for Pertioned Main Memory Databases Disponible en: http://db.csail.mit.edu/pubs/hstore-cc.pdf

Netezza Corporation. 2009. Netezza database users guide
Disponible en:
http://www.enzeecommunity.com/servlet/JiveServlet/previewBody/1161-102-1-1107/Netezza_database_users_guide.pdf

Nevado. M.V. 2010. *Introducción a las bases de datos relacionales*. Madrid: Vision Libros.

Null L. and Lobur J. 2006. The essentials of computer organization and architecture. n.p.: Jones and Bartlett. pp. 741-742.

Ramírez Galeano, Edgar ejecución de consultas y marcado masivo en el dbb . Colegio de postgraduados. 2006

Reena Mathews. Simple Strategies to Improve Data Warehouse Performance. North Carolina State University. 2004

Rob P. y Coronel C. 2009. Database Systems: Design. Implementation. and Management. Boston. MA: Course Technology. pp. 90-91.

Salam. F. and Stevens.J.R. 2007. Semantic Web Technologies and E-Business: toward the integrated virtual organization and business process automation. Hershey. PA: Idea Group. pp. 11-12. 2007.

Shapiro J. 2006. Microsoft SQL Server 2005: the complete reference. NY: Mc Graw Hill. pp. 54-55.

Stavros Harizopoulos, 2008. Samuel Madden, Michael Stonebraker, Daniel J. Abadi. OLTP Through the Looking Glass, and What We Found There. ACM: 978-1-60558-102-6/08/06 disponible en: http://hstore.cs.brown.edu/papers/hstore-lookingglass.pdf

Strohm Richard. Et. Al. 2008. Database Concepts, 11 g Release 1(11.1). Disponible en: http://download.oracle.com/docs/cd/B28359_01/server.111/b28318.pdf

Teklitz Frank. 2000. Sybase Business Intelligence and Data Warehousing Solutions for Sybase IQ. Disponible en: http://www.sybase.com/content/1010008/rcubes_wp_L01496-v2.pdf

The PostgreSQL Global Development Group. 2009. PostgreSQL 8.4 Official Documentatio. Volume V. Linbrary. pp. 161-162.

TranQuery Q. T. 2000. Processing Using Compressed Bitmaps. (Oct. 31).

Vertica Systems Inc. 2010. Fast Data Loading into the Vertica Analytic Database "Vertica data loading architecture, benchmarks and best practice" disponible en: http://www.vertica.com/wp-content/uploads/2011/01/FastDataLoadingInVertica.pdf

Weiss M. A. 1993. Data structures and Algorithm Analysis. Reedwodd City. CA:

Wikipedia. 2010. Tag metadata [Online]. Available: http://en.wikipedia.org/wiki/Tag_%28metadata%29

Wikipedia. (2010). Big Table [Online]. Available: http://en.wikipedia.org/wiki/BigTable

Wikipedia. (2010). Cardinality [Online]. Available: http://en.wikipedia.org/wiki/Cardinality

ANEXO (CD)

En el disco compacto que se entrega junto con la tesis (o en el mismo disco en caso de ser la versión digital de la tesis) se incluyeron los siguientes materiales:

1. El esquema de la base de datos en SQL SERVER.
2. La base de datos DATOSGENBASE, completa (con los datos.)
3. El esquema de la base de resultados obtenidos (tiempos de respuesta.) No se incluyen los datos puesto que ya fueron incluidos en el cuerpo de la tesis.
4. El programa fuente en Visual Basic que se usó para generar los datos.
5. El programa con el que se efectuaron las consultas en SQL (aunque en algunos casos, se hizo directamente usando comandos de SQL “en vivo”.
6. El programa fuente que se usó para cargar los datos a las estructuras del DBB y para efectuar las consultas en esta tecnología.