ESCUELA SUPERIOR DE INFORMÁTICA
(Ciudad Real)

Asignatura: Bases de Datos

Título del trabajo:
DISEÑO FÍSICO DE LA BASE DE DATOS

Trabajo realizado por:
Mª teresa fernández novas

Profesor:
Francisco ruíz

UNIVERSIDAD DE CASTILLA - LA MANCHA

Fecha: 6-4-2000

INDICE:

- Documentos elegidos para la organización de ficheros.

4.3.- Añadir índices secundarios.
- Objetivo.
- Documentos elegidos para los índices secundarios.

4.4.- Considerar la introducción de los controles de redundancia.
- Objetivo.
- Denormalización.
- Contrareferencias de transacciones y relaciones.
- Ejemplos.

4.4.1.- Considerar datos derivados.
- Ejemplos.

4.4.2.- Considerar atributos duplicados o juego de relaciones juntas.
- Tipos de denormalización. Ejemplos.
- Circunstancias en las que se debería considerar la denormalización.
- Introducción de documentos redundantes.
- Ejemplos.

4.5.- Estimar el espacio del disco requerido.
- Objetivo.
- Ejemplos.

5.- Diseño de los mecanismos de seguridad.

- Objetivo.

5.1.- Diseño según la perspectiva del usuario.
- Objetivo.
- Ejemplos.

5.2.- Diseño de normas de acceso.
- Objetivo.
- Privilegios.
- Documentos diseñados para vistas de usuario y medidas de seguridad.
- Ejemplos.

6.- Monitorización y sintonización del sistema operacional.

- Objetivo.
- Beneficios.
- Tipos.

7.- Elementos de Diseño Físico.

7.1.- Índice compuesto relacional.

7.2.- Área de datos.
- Definición de clave nativa.
- Características.

8.- ORACLE 8

8.1.- Introducción al servidor Oracle

8.1.1 Bases de Datos y dirección de la información.
8.1.2 Estructura de la Base de Datos y espacio de la dirección.
8.1.3 Estructura de la Memoria y Procesado.
8.1.4 Concurrencia de los datos y consistencia.
8.1.5 Procesamiento distribuido y Bases de datos distribuidas.
8.1.6 Comenzar y acabar las operaciones.
8.1.7 Seguridad de la Base de Datos.
8.1.8 Bases de Datos de reserva y recuperación.
8.1.9 El modelo relacional para la dirección de la Base de Datos.
8.1.10 Acceso a datos.

PREFACIO

El diseño de Bases de Datos es el proceso por el que se determina la organización de una Base de Datos. El diseño de una Base de Datos se realiza generalmente en tres fases:

La primera fase es el diseño conceptual.
La segunda fase se refiere al diseño lógico.
Y la última fase es la que estudiaremos en particular, es el Diseño Físico.

El trabajo se divide en tres partes, precedidas por un capítulo introductorio al Diseño de la Base de Datos y los objetivos que persigue, el segundo capítulo es introductorio al Diseño Físico de la Base de Datos y el tercer capítulo abarca todo lo relacionado con el Diseño Físico de Bases de Datos, desde la metodología del diseño físico, representación, mecanismos de seguridad, y elementos del diseño físico.

Se incluye también un resumen de todo lo relacionado y mencionado en todos estos capítulos.

Capítulo de Objetivos

En el capítulo del Diseño Físico de Bases de Datos estudiaremos:

- El propósito del Diseño Físico de la Base de Datos.
- Como trazar el mapa del diseño lógico de la Base de Datos a un Diseño Físico de la Base de Datos.
- Como diseñar las relaciones base para la tarjeta DBMS; DataBase Management Systems. (Sistemas de Gestión de Bases de Datos).
- Como diseñar la fuerza de la empresa, para la tarjeta DBMS.
- Como seleccionar el fichero en organizaciones apropiadas, basadas en el análisis de transacciones.
- Cuando usar los índices secundarios para mejorar el funcionamiento.
- Cuando denormalizar para mejorar el funcionamiento.
- Como estimar el tamaño de la Base de Datos.
- Como diseñar los mecanismos de seguridad para satisfacer los requerimientos del usuario.
- La importancia de la monitorización y sintonización del sistema operacional.

RESUMEN

Como **resumen** de todo este capítulo dedicado al Diseño de la Base de Datos Física podríamos decir que el Diseño de la Base de Datos Física es el proceso de producir una descripción de la implementación de la Base de Datos en un almacenamiento secundario. Describe las relaciones base y el almacenaje de estructuras y métodos de acceso usados para acceder a los datos de forma efectiva. El diseño de las relaciones base puede ser acometidas solo una vez que el diseñador es totalmente consciente de las facilidades ofrecidas por la tarjeta DBMS.

La finalidad del Diseño Físico es encontrar una estructura interna que soporte la estructura conceptual y los objetivos del diseño lógico con la máxima eficiencia de los recursos máquina.

El paso inicial del diseño de la Base de Datos Física es la traducción del modelo de datos lógico global en una forma que puede ser implementada en la tarjeta relacional DBMS.

El siguiente paso, diseña el fichero de organización y métodos de acceso que puedan usarse para almacenar. Esto implica analizar las transacciones que se ejecutaran en la Base de Datos, elección conveniente de ficheros de organización basados en estos análisis, añadir índices secundarios, introducir y controlar la redundancia para mejorar el funcionamiento, y finalmente estimar el espacio del disco que será necesario para la implementación.

Los ficheros *Heap*, son buenos para insertar un gran número de registros en el fichero. Son inapropiados cuando solo seleccionamos registros que son recuperados. *Hash* son ficheros buenos cuando la recuperación está basada en una clave exacta. Esta técnica permite determinar la localización física de una determinada fila directamente sin necesidad de usar índices. De este modo podremos recuperar los datos con una sola lectura. No son buenas cuando la recuperación está basada en maquinas paternas, valores de rango, claves compartidas, o cuando la recuperación está basada en un atributo del fichero Hash.

ISAM es una estructura versátil de estructuras de almacenamiento que Hashing. Ello supone recuperaciones basadas en un juego de claves exactas, rango de valores, y la especificación de claves compartidas. De esta manera el índice ISAM es estático, el funcionamiento del fichero ISAM se deteriora cuando las relaciones son actualizadas.

Los *árboles* B^+ son ficheros de organización, con índices dinámicos. Una organización para los índices proporciona un rápido acceso a los datos, basándose en el contenido de una clave, con el pequeño costo que supone mantener los índices actualizados, los árboles B^+ contienen los valores de las claves. La comprensión de los índices abarca el almacenamiento de las clav4s una sola vez, independientemente de cuántas veces aparece en las filas de una tabla, así como el almacenamiento únicamente de aquellas porciones clave que difieren de las claves anteriores.

Los *índices secundarios* proporcionan un mecanismo para especificar una clave adicional para una relación base que puede ser usada de forma más eficiente. Esto implica que el mantenimiento y uso de los índices secundarios tienen que ser balanceados para el perfecto funcionamiento.

INGRES comprende una serie de módulos o herramientas de trabajo, tanto para usuario final como para desarrollo. Esta herramienta incluye los lenguajes de consulta SQL y QUEL. El diccionario de datos activo está integrado con la estructura de la Base de Datos INGRES. El diccionario es una Base de Datos que se puede consultar. A medida que creamos tablas, listados, programas o gráficos el diccionario irá guardando toda la información de forma automática, sin que el usuario tenga que hacerlo. Al ser un BD INGRES, un usuario con cierto privilegio, podrá consultar, e incluso sacar referencias de la información existente, y en un momento dado incluso actualizarla. Existe el módulo INGRES / VISION que es una

herramienta que facilita el desarrollo de aplicaciones. Se define de manera visual las pantallas y procedimientos asociados a esas aplicaciones para acceder a la BD.

VISION permite ir desarrollando la aplicación según se va respondiendo a las opciones de menú para dar instrucciones INGRES o para especificarle como construir la aplicación. No es necesario conocer como es el código para poder ejecutar la aplicación.

1.- DISEÑO DE LA BASE DE DATOS

1.1. - Introducción al Diseño de la Base de Datos

Las fases de creación de una Base de Datos, es una operación difícil, larga y costosa. Por ello la responsabilidad de las decisiones no solo repercute en el informático sino también en los directivos y los usuarios que también tienen protagonismo.

La puesta en marcha de una Base de Datos está compuesta por las siguientes fases:

1) *Estudio previo y plan de trabajo –estrategia--.*
2) *Concepción de la Base de Datos y selección del equipo.*
3) *Diseño y carga.*
4) *Producción.*

El punto 1 (*Estudio previo y plan de trabajo –estrategia--*); se refiere al análisis previo, estudio de la viabilidad, este punto debe preceder siempre a la fase del diseño de una Base de Datos; en el análisis se deben implicar los directivos, definiendo unos objetivos claros y concretos que sirvan de pauta en todo el desarrollo. El estudio de la viabilidad es corto de duración y en la cuál los técnicos intervienen poco, es fundamental para conseguir que el nuevo sistema de información se integre en el organismo, adaptándose a sus objetivos y prestando los servicios que de él se esperaban.

Una vez que la dirección ha tomado la decisión inicial de emprender las operaciones que conducen al establecimiento de un sistema de Base de Datos y se han definido los objetivos del proyecto pasamos al punto 2 (*Concepción de la Base de Datos y selección del equipo*); donde la responsabilidad del proceso de creación de la BD recae generalmente sobre el administrador de la base, el cual en colaboración con los usuarios deberán conseguir una estrategia efectiva de los datos para conseguir los siguientes objetivos:

- Facilitar a los usuarios de la base la obtención de los datos para cumplir sus obligaciones dentro de la empresa.
- Conseguir que esos datos estén disponibles en un plazo razonable que permita apoyar en ellos la toma de decisiones.
- Optimizar el rendimiento de los recursos dedicados al sistema.

El punto 3 (*Diseño y carga*); está basado en la definición del diseño lógico y físico, una vez que se ha obtenido el esquema conceptual en el punto anterior. La carga se refiere a que una vez definida la estructura física de la Base de Datos se ha de proceder a cargar los datos en la misma. Una vez cargados en la base algunos ficheros, se debe comenzar a realizar las pruebas de la Base de Datos y medir sus rendimientos, con objeto de ir ajustando la estructura física e incluso, a veces, la estructura lógica con el fin de optimizar.

Estas fases son una aproximación al:

- *Diseño conceptual:* cuyo objetivo es obtener una buena representación de los recursos de información de la empresa, con independencia de usuarios o aplicaciones en particular, y fuera de consideraciones sobre la eficiencia del ordenador.
- *Diseño lógico:* cuyo objetivo es transformar el esquema conceptual obtenido en la etapa anterior, adaptándolo al modelo de datos en el que se apoya el SGBD que se va a utilizar.
- *Diseño Físico:* cuyo objetivo es conseguir una instrumentación, lo más eficiente posible, del esquema lógico.

Cabe destacar que cada fase es un proceso iterativo, y que se van produciendo refinamientos sucesivos antes de pasar a las siguientes fases.

Existe una realimentación entre el diseño lógico y el físico, ya que pueden producirse cambios en el diseño lógico derivados de requisitos del Diseño Físico; es decir, muchas veces es preciso adaptar el diseño lógico para conseguir una mayor eficiencia del sistema. Así mismo, no sería conveniente que existiese una realimentación del diseño lógico y físico hacia el diseño conceptual, ya que éste debe representar los recursos de información de la empresa con independencia de aspectos técnicos. Las fases del diseño de la Base de Datos se pueden relacionar con las fases del diseño de un *sistema de información:*

- *El análisis funcional* integra el diseño conceptual, en el que, a partir de los requisitos de información, se produce el esquema conceptual.
- *El análisis orgánico* integra los diseños lógico y físico. Así mismo, a partir del esquema conceptual, y considerando el modelo de datos en el que se basa el SGBD y los requisitos de los procesos se obtiene:

  - *El esquema lógico global*: es decir, el esquema global de la BD, en el modelo relacional, Codasyl o Jerárquico propio del SGBD.
  - *Vistas de usuario*: son aquellas estructuras externas derivadas del esquema lógico global que resulten de mayor interés en la utilización del sistema.

El conjunto de datos y sus interrelaciones, que aparecen en el esquema lógico, han de ser estructurados y almacenados en un determinado dispositivo físico, operación que conviene separar en dos fases: en la primera se obtiene el llamado esquema interno o almacenado, y en la segunda se procede a pasar dicha estructura a un soporte físico concreto, obteniéndolo se la Base de Datos Física.

El Administrador de la Base de Datos debe tener un instrumento para poder definir, a partir del esquema lógico, la representación de la Base de Datos en el almacenamiento, a fin de privilegiar caminos de acceso que mejoren los rendimientos y cumplir los objetivos operacionales de la Base de Datos. Se deberán desarrollar los programas y procedimientos necesarios de forma que cuando se vaya cargando en la Base de Datos los distintos conjuntos de información se puedan ir probando los programas que manejan esos datos. Una vez que algunos ficheros están cargados en la Base de Datos, se deben comenzar las pruebas de la Base de Datos y medir sus rendimientos, con objeto de poder ir ajustando la estructura física.

En el proceso del diseño de una Base de Datos se deben especificar las *entradas* y *salidas*. Para realizar el proceso de entrada debemos definir la información y los objetivos que se han especificado al plantear el diseño de la Base de Datos; esto vendrá determinado por las entrevistas realizadas al usuario y el análisis de los documentos a generar (listados, pantallas…), junto con los objetivos de la empresa.

Otros procesos de entrada importantes a definir en el diseño de la Base de Datos son:

- *Requisitos de proceso*; es decir, las distintas características que deben cumplir los programas o aplicaciones que actúen sobre la BD (por ejemplo, el tiempo de respuesta).
- *Especificaciones del SGBD (Sistema de Gestión de Base de Datos)*; que incluirán el modelo de datos soportado, además de las características de rendimiento, seguridad, lenguajes… también habrá que estudiar las herramientas y métodos que nos ayudarán a diseñar el diseño lógico y físico de la Base de Datos.
- *Configuración del equipo físico y del SO (Sistema Operativo)*: que influirá en mayor o menor medida en el desarrollo de la Base de Datos, así como en la etapa del Diseño Físico.

En el proceso de salida habrá que indicar:

- *Estructuras lógicas de datos:* como resultado del proceso de diseño se obtendrá el esquema conceptual de la base, el esquema lógico en el modelo soportado por el SGBD, y algunas de las principales vistas de usuario que se precisen para interactuan con la BD.

- *Estructuras de almacenamiento:* es el esquema interno donde aparecerán especificados los parámetros y aspectos de Diseño Físico del sistema, como particiones, definiciones de espacio, índices, agrupamientos, ….
- *Normativa de explotación:* donde se incluirán los aspectos de confidencialidad, seguridad e integridad para la explotación y el mantenimiento de la base.
- *Especificaciones para los programas de aplicación:* para los que se determinan ciertas características a cumplir, especialmente en lo que se refiere al mantenimiento de la integridad, confidencialidad, y seguridad de la base que no puedan ser recogidas en el esquema.

Los niveles de abstracción de la arquitectura ANSI facilitan el diseño de una Base de Datos, al proporcionarnos instrumentos que ayudan a la estructuración del mundo real hasta llegar a la Base de Datos Física. El problema se encuentra en que todavía no hay un modelo conceptual admitido e instrumentado en los SGBD, no se puede pasar de forma automática de una estructura conceptual a una estructura física, teniéndose que recorrer el camino en dos etapas:

1) Primera etapa; consiste en adaptar la estructura conceptual a las exigencias y restricciones del modelo de datos propio del SGBD que vaya a ser utilizado.
2) La segunda etapa; es que habrá que obtener el esquema almacenado mediante especificaciones que tiendan a conseguir la máxima eficiencia de cara a la máquina y al problema concreto.

El mundo de los datos son las cadenas de caracteres a las que nos referíamos anteriormente, y que carecen de significado, si no disponemos de los medios necesarios para recorrer el camino inverso, pasando al mundo real con ayuda del lenguaje de manipulación, que nos permitirá recuperar los datos almacenados en la base, reincorporándoles su contenido semántico y obteniendo la información que necesita el usuario.

Aplicando al esquema conceptual las reglas del modelo de datos propio del SGBD que se va a utilizar, se obtiene la estructura lógica global, y de ésta se pasa al esquema interno. Las Base de Datos física se ha de apoyar en los métodos de acceso del sistema operativo, y se ha de rellenar con los valores que se obtienen por observación de los sucesos del mundo real (ocurrencias o instancias).

Las herramientas CASE están proporcionando en estos momentos una importante ayuda en el diseño de bases de datos, al disponer de modelos de datos semánticos, que facilitan el diseño conceptual y realizan la transformación al esquema relacional propio de los productos comerciales más extendidos.

1.2. – Objetivos del diseño de la Base de Datos.

- Profundizar en la semántica ligada a los modelos de datos e interrelacionarlos con el ciclo de vida de un sistema de información.
- Aportar conceptos para el diseño de BD.
- Profundizar en el modelo E/R de Chen como soporte conceptual para la modelización de BD.
- Analizar el ciclo de diseño de una BD, contemplando las transformaciones necesarias para llegar desde un modelo conceptual a un modelo de implementación en un SGBD comercial.
- Describir la funcionalidad de los módulos o subsistemas más significativos que configuran un SGBD.

2.- DISEÑO FÍSICO DE LA BASE DE DATOS

2.1.- Introducción al Diseño Físico de la Base de Datos

El punto de partida del Diseño Físico de la Base de Datos es el modelo de datos de lógica global y la documentación que describe el modelo creado en los pasos 2 y 3 de la metodología del diseño de la Base de Datos lógica.

La metodología comienza por refinar los modelos conceptuales locales creados en el paso 1, en los modelos de datos lógicos locales y después usa los modelos lógicos para derivar un grupo de relaciones.

Los modelos lógicos y las relaciones derivadas fueros validadas usando la técnica de normalización, y contra las transacciones que ellos deben soportar para los usuarios. La fase del diseño de la Base de Datos lógica fué concluida por la fusión de los modelos de datos locales (que representa el criterio de cada usuario de la empresa) junto a la creación de un modelo de datos global (que representa todos los criterios del usuario de la empresa).

En la tercera y final fase de la metodología del diseño de la Base de Datos, el diseñador debe decidir como trasladar el diseño de la Base de Datos lógica (que es, las entidades, atributos, relaciones y fuerzas) a un diseño de la Base de Datos física que puede ser implementada usando la tarjeta DBMS. Como muchas partes de Diseño Físico de la Base de Datos son altamente dependientes de la tarjeta DBMS, debe haber más de una forma de implementar alguna parte dada de la Base de Datos. Consecuentemente, el diseñador debe ser completamente consciente de la funcionalidad de la tarjeta DBMS, y debe comprender las ventajas y desventajas de cada alternativa para una implementación particular.

El diseñador también debe ser capaz de seleccionar una estrategia conveniente de almacenamiento que tenga en cuenta el uso.

En este tema demostramos como convertir las relaciones derivadas del modelo de datos lógico global en una implementación de la Base de Datos específica. Proporcionamos los principios para cambiar estructuras de almacenamiento por relaciones de base, decidiendo cuando crear índices, y cuando denormalizar el modelo de datos lógico e introducir desempleo.
Más adelante, mostramos los detalles de implementación física para aclarar la discusión. Antes presentamos la metodología para diseños de la Base de Datos física, brevemente repasamos los procesos del diseño.

2.2.- Comparación del Diseño Físico de la Base de Datos y el Diseño Lógico.

En la presentación de la metodología del diseño de la Base de Datos, dividimos el proceso del diseño en tres fases principales:

1) ***Diseño de la Base de Datos Conceptual.***
2) ***Diseño de la Base de Datos Lógica.***
3) ***Diseño de la Base de Datos Física.***

La fase anterior al Diseño Físico, fué el diseño de la Base de Datos Lógica, es en gran parte independiente de los detalles de implementación, tal como el funcionamiento específico de la tarjeta DBMS, y la aplicación de los programas, pero es dependiente en el modelo de datos de la tarjeta. El rendimiento de este proceso es un modelo y documentación de datos lógico y global que describe este modelo, así como el diccionario de datos y el esquema relacional. A la vez, estos representan las fuentes de información para el proceso del Diseño Físico, y proporcionan al diseñador la Base de Datos física, con un vehículo para hacer los empleos desconectados que son tan importantes para un diseño de la Base de Datos eficiente.

Mientras el diseño de Base de Datos lógico se interesa del ´´qué´´, el diseño de la Base de Datos Física se interesa por el ´´cómo´´. Esto requiere diferentes destrezas que son a menudo fundadas en personas diferentes. En particular, el diseño de la Base de Datos física debe conocer como funciona el sistema huésped del ordenador del DBMS, y debe darse completa cuenta del funcionamiento de la tarjeta DBMS. Mientras el funcionamiento proporcionado por sistemas

corrientes muy variados, el Diseño Físico debe estar hecho a medida a un sistema específico DBMS.

Sin embargo, el Diseño Físico de la Base de Datos no es una actividad aislada, hay a menudo Feed-Back entre el Diseño Físico, lógico y de aplicación. Por ejemplo, las decisiones tomadas durante el Diseño Físico para mejorar el funcionamiento podría afectar a la estructura del esquema lógico.

3.- PERSPECTIVAS DE LA METODOLOGÍA DEL DISEÑO FÍSICO DE LA BASE DE DATOS

3.1.- Diseño Físico de la Base de Datos: es el proceso de producción de una descripción, de una implementación, de un almacenamiento secundario de la Base de Datos, describe el almacenamiento de estructuras y métodos de acceso usados para conseguir el acceso eficiente a los datos. El Diseño Físico de la Base de Datos es la última etapa del proceso de diseño, en el cual, teniendo presentes los requisitos de los procesos, características del SGBD, del SO y el hardware, se pretenden los siguientes objetivos:

- **Disminuir los tiempos de respuesta.**
- **Minimizar espacio de almacenamiento.**
- **Evitar las reorganizaciones.**
- **Proporcionar la máxima seguridad.**
- **Optimizar el consumo de recursos.**

En definitiva lo que se pretende alcanzar es el cumplimiento de los objetivos de sistema y conseguir optimizar el ratio coste/beneficio.

La poca flexibilidad de los sistemas comerciales obliga a llevar a cabo la reestructuración de las relaciones para conseguir tiempos de respuesta aceptables. Por tanto, se deberá proceder de forma iterativa desde el diseño lógico al Diseño Físico, y viceversa para poder conseguir el ratio anteriormente citado.

Así mismo, no existe un modelo formal para el Diseño Físico (como por ejemplo, el modelo relacional para el diseño lógico), el Diseño Físico resulta muy dependiente del producto comercial concreto hasta el momento.

El Diseño Físico consta de entradas y salidas. En las entradas se podría destacar además de los objetivos del Diseño Físico; los recursos máquina (soporte físico), recursos lógicos (sistemas operativos), esquema lógico y la información sobre las aplicaciones (tiempos de respuesta y seguridad).

A partir de las entradas, en la salida obtendremos; normas de seguridad, estructura interna, y especificaciones para el ajuste.

El problema del Diseño Físico para el administrador de la Base de Datos consiste en proveer un conjunto eficiente de estructuras de acceso de modo que el optimizador pueda tomar las mejores decisiones. Entre los instrumentos más importantes del Diseño Físico se encuentra la selección de los índices secundarios, que es uno de los problemas en la instrumentación física de una Base de Datos.

Una vez diseñadas las aplicaciones se conocerá cuales son las consultas más frecuentes y prioritarias a la Base de Datos, por lo que será conveniente crear un índice secundario que ayude a localizar las filas seleccionadas en dichas consultas y reducir los accesos a disco. La forma más usual para crear un índice en SQL es:

```
CREATE [UNIQUE] INDEX <nombre del índice>
ON <nombre de la tabla> <parent. izquierdo> <lista de columnas> <parent. derecho>
```

Así con la cláusula UNIQUE podemos utilizar los índices para garantizar que los datos de una columna no se repiten. Por ejemplo si tenemos una relación LIBRO (CODIGO, TITULO, IDIOMA, EDITORIAL) y el mayor número de consultas se hace preguntando por el idioma, se puede crear un índice como el que sigue:

```
CREATE INDEX idiomalibro
        ON Libro (idioma)
```

Existen otros elementos importantes en el Diseño Físico, aunque no todos los sistemas comerciales disponen de ellos, y si existen el diseñador tiene la posibilidad de actuar sobre ellos

ajustándolos a cada caso concreto, algunos de ellos se muestran a continuación, aunque más adelante los podremos ver con más detalle:

- Registros físicos.
- Punteros.
- Direccionamiento calculado (Hashing)
- Agrupamientos (cluster)
- Bloqueo y comprensión de datos.
- Asignación de espacios de almacenamientos como memorias intermedias (buffers).
- Asignación de conjuntos de datos a particiones y a dispositivos físicos.

. . .

En general los fabricantes muestran tres estrategias de Diseño Físico:

1) El SGBD *impone* una estructura interna, dejándole al diseñador muy poca flexibilidad, lo que suele aumentar la independencia física/lógica, pero disminuye la eficacia.
2) El administrador *diseña* la estructura interna, esto supone una importante carga para el administrador y puede influir de forma negativa en la independencia, aunque puede mejorar la eficacia.
3) El SGBD *proporciona* una estructura interna *opcional* que el diseñador puede cambiar a fin de optimizar el rendimiento de la Base de Datos. Esta estrategia tiene unas ventajas:

    - La Base de Datos puede comenzar a funcionar de inmediato.
    - La eficacia va aumentando al irse efectuando los ajustes sucesivos.
    - La independencia se mantiene.

A continuación describiremos los pasos de la metodología del Diseño Físico de la Base de Datos:

1.- Diseño Físico de la Base de Datos.
2.- Traducción del modelo de datos lógico global para tarjetas DBMS.
3.- Diseño de las bases relacionales para la tarjeta DBMS.
4.- La fuerza de la empresa para diseñar la tarjeta DBMS.
5.- Representación del Diseño Físico.
6.- Análisis de transacciones.
7.- Seleccionar el fichero en las organizaciones.
8.- Seleccionar índices secundarios.
9.- Considerar la introducción de los controles desempleados.
    9.1: Considerar datos derivados.
    9.2: Considerar atributos duplicados o juego de relaciones juntas.
10.-Estimar el espacio del disco requerido.
11.-Diseño de los mecanismos de seguridad.
    11.1 Diseño de las perspectivas de los usuarios.
    11.2 Diseño de las reglas de acceso.
12.-Monitorización y sintonización del sistema operacional.

La metodología del Diseño Físico de la Base de Datos presentada está dividida en cuatro pasos (anteriormente citados) principales numerados consecutivamente.

El Diseño Físico de la Base de Datos implica el diseño de las relaciones de la Base y se integra fuertemente usando el funcionamiento disponible de la tarjeta DBMS.

Traducción del modelo de datos lógico global para tarjetas DBMS; implica seleccionar las estructuras de almacenamiento y los métodos de acceso para las relaciones base.

Típicamente, el DBMS, implica un número de alternativas para construir un almacén de datos, con la excepción del PC DBMS, el cual tiende a ajustar el almacenamiento construido. Desde el punto de vista de los usuarios, la representación del almacenamiento interno para las relaciones debería ser transparente – el usuario debería poder acceder a las relaciones y tuplas sin tener que especificar donde o como las tablas están almacenadas.

Esto requiere que el DBMS proporcione datos físicos independientes para que los usuarios no se vean afectados por cambios de la estructura física de la Base de Datos, como se discutió anteriormente.

El trazado entre el modelo de datos lógico y el modelo de datos físico está definido en el esquema interno. El diseñador debe proporcionar los detalles del Diseño Físico para ambos, el DBMS y el sistema operativo. Para el DBMS, el diseñador debe especificar las estructuras de fichero que son usados para representar cada relación; Para el sistema operativo, el diseñador debe especificar los detalles tales como la localización y protección de cada fichero. El paso 2 también considera enervante (es decir, debilitar las fuerzas físicas) la fuerza de normalización impuesta sobre el modelo lógico de datos para proporcionar todo el funcionamiento del sistema. Este es un paso que solo se acometerá si fuera necesario, porque los problemas inherentes implican la introducción del desempleo, mientras mantengan su consistencia.

El diseño de las Bases Relacionales para la tarjeta DBMS; implica el diseño de medidas de seguridad para proteger los datos desde un acceso inautorizado. Esto implica decidir como cada modelo lógico global de datos debería estar implementado, y los controles de acceso que son requeridos en las relaciones de base.

La fuerza de la empresa para diseñar la tarjeta DBMS; es un proceso continuo de monitorización del sistema operacional para identificar y resolver cualquier problema del funcionamiento resultante del diseño, e implementación de nuevos o cambiantes requerimientos.

3.2.- La Metodología del Diseño Físico de la Base de Datos para la Base de Datos Relacional.

Esta sección nos proporciona una guía paso a paso para introducir el Diseño Físico de la Base de Datos para la Base de Datos relacional. Por tanto, en esta metodología, demostramos la asociación cerrada entre el Diseño Físico de la Base de Datos y la implementación para describir como los diseños alternativos pueden implementarse usando varias tarjetas DBMS.

3.2.1: Traducción del modelo de datos lógico global para tarjetas DBMS

OBJETIVO: Producir un esquema de trabajo básico de la Base de Datos relacional desde el modelo de datos lógico global.

La primera actividad del Diseño Físico de la Base de Datos implica la traducción de las relaciones derivadas del modelo de datos lógico global dentro de una forma que puede implementarse en la tarjeta relacionada DBMS.

La primera parte de este proceso supone cotejar (es decir, confrontar una cosa con otra),la información recogida durante el modelado y documentación de los datos lógicos en el diccionario de datos.

La segunda parte de este proceso usa esta información para producir el diseño de la base relacional. Este proceso requiere un conocimiento profundo de las funciones ofrecidas por la tarjeta DBMS. Por ejemplo, el diseñador necesitará conocer:

- Si el sistema soporta la definición de clave primaria, clave secundaria y clave externa.
- Si el sistema soporta la definición de datos requeridos (que es, si el sistema permite atributos definidos como NO NULOS).
- Si el sistema soporta la definición de dominios.
- Si el sistema soporta la definición de la fuerza de la empresa.
- Como crear una base relacional.

La traducción del modelo de datos lógico global para tarjetas DBMS incluye dos actividades:

3.2.1.1: Diseño de las bases relacionales para la tarjeta DBMS.
3.2.1.2: La fuerza de la empresa para diseñar la tarjeta DBMS.

3.2.1.1 Diseño de las Bases Relacionales para la Tarjeta DBMS.

OBJETIVO: Para decidir como representar las bases relacionales, nosotros vamos a identificar el modelo de datos lógicos global en las tarjetas DBMS.

Para comenzar el proceso del Diseño Físico, primero necesitamos cotejar y asimilar la información sobre relaciones producidas durante el modelaje de datos lógicos. La información puede ser obtenida desde el diccionario de datos y la definición de las relaciones definidas usando el Database Design Language (DBDL). Por cada relación identificada en el modelo de datos lógico global, nosotros vamos a definir los siguientes pasos:

- El nombre de la relación.
- Una lista de los atributos simples que soporta.
- La clave primaria y, cuando sea apropiado, las claves alternativas (AK), y las claves externas (FK).
- Integrar la fuerza de alguna clave externa identificada.

Desde el diccionario de datos, también tenemos para cada atributo:

- Los dominios, la consistencia de un tipo de dato, longitud, y alguna fuerza en el dominio.
- Una opción para dejar de evaluar los atributos.
- Si el atributo puede tener nulidad.
- Si el atributo es derivado y, como sería computado.

Para representar el diseño de la base relacional, usaremos el DBDL para definir los dominios, dejar de evaluar, y los indicadores nulos. Por ejemplo, para la relación Propietario_de la_Renta de la *DreamHome* caso de estudio, debemos producir el diseño que mostramos a continuación:

Dominio número propietario: longitud variable, cadena de caracteres de longitud 5
Dominio calle: longitud variable, cadena de caracteres como máximo longitud 25
Dominio área: longitud variable, cadena de caracteres como máximo longitud 15
Dominio ciudad: longitud variable, cadena de caracteres como máximo longitud 15
Dominio Post_codigo: longitud variable, cadena de caracteres como máximo longitud 8
Dominio tipo de propietario: carácter simple ´B´,´C´,´D´,E´,´F´,´M´,´S´
Dominio habitación del propietario: entero, en el rango de 1 a15
Dominio renta del propietario: valor monetario, en el rango 0.00-9999.00
Dominio número del dueño: longitud variable, cadena de caracteres de longitud 5
Dominio nº de personal en plantilla: longitud variable, cadena de caracteres de longitud 5
Dominio número de sucursal: longitud variable, cadena de caracteres de longitud 3

Propiedad_de la_ Renta (				
	Pno:	número propietario	NOT NULL,	
	Calle:	calle	NOT NULL,	
	Area:	área,		
	Ciudad:	ciudad	NOT NULL,	
	Pcode:	Post-codigo,		
	Tipo:	tipo de propietario	NOT NULL	DEFAULT: ´F´,
	Habitación:	habitación del propietario	NOT NULL	DEFAULT: 4,

Renta:	propietario de la renta	NOT NULL	DEFAULT: 600,
Ono:	número del propietario	NOT NULL,	
Sno:	nº de personal en plantilla,		
Bno:	número de sucursal	NOT NULL).	

PK (Clave primaria): Pno
FK (Clave externa): Son REFERENCIAS personal (Sno) modo de borrado SET NULL, modificación CASCADE
FK: Ono REFERENCIAS propietario (Ono) modo de borrado NO ACTION, modificado CASCADE.
FK: Bno REFERENCIAS sucursal (Bno) modo de borrado NO ACTION, modificado CASCADE.

El siguiente paso es decidir como implementar las bases relacionales. Esta decisión es dependiente de la tarjeta DBMS. Algunos sistemas proporcionan más facilidades que otros para definir las bases relacionales y la fuerza de integración. Para ilustrar este proceso, mostraremos cuatro formas principales de crear relaciones y el uso de la fuerza de integración:

1) El 1992 ISO SQL estándar (SQL 92).
2) Desencadenamiento.
3) INGRES 6.4.
4) Indexación única.

El ISO SQL 1992 estándar (SQL 92)

Si la tarjeta DBMS es completa con el 1992 ISO SQL estándar, entonces es relativamente fácil diseñar la implementación de la base. Por ejemplo, al crear la relación Propietario_de la_Renta podríamos usar el SQL de la siguiente manera:

```
CREATE DOMAIN número propietario AS VARCHAR (5)
        CHECK (VALUE IN (SELECT Ono FROM propietario))
CREATE DOMAIN número de personal en plantilla AS VARCHAR (5)
        CHECK (VALUE IN (SELECT Son FROM personal en plantilla))
CREATE DOMAIN número de sucursal AS VARCHAR (3)
        CHECK (VALUE IN (SELECT Bno FROM sucursal))
CREATE DOMAIN número del dueño AS VARCHAR (5)
CREATE DOMAIN calle AS VARCHAR (25)
CREATE DOMAIN área AS VARCHAR (15)
CREATE DOMAIN ciudad AS VARCHAR (15)
CREATE DOMAIN post_codigo AS VARCHAR (8)
CREATE DOMAIN tipo de propietario AS CHAR (1)
        CHECK (VALUE IN (´B´,´C´,´D´,E´,´F´,´M´,´S´))
CREATE DOMAIN habitación del propietario AS SMALLINT
        CHECK (VALUE BETWEEN 1 AND 15)
CREATE DOMAIN renta del propietario AS DECIMAL (6,2)
        CHECK (VALUE BETWEEN 0 AND 9999)

CREATE TABLE Propietario_de la_Renta (
        Pno             número propietario          NOT NULL,
        Calle:          calle                       NOT NULL,
        Area:           área,
        Ciudad:         ciudad                      NOT NULL,
        Pcode:          Post-codigo,
        Tipo:           tipo de propietario         NOT NULL    DEFAULT: ´F´,
        Habitación:     habitación del              NOT NULL    DEFAULT: 4,
```

	propietario		
Renta:	propietario de la renta	NOT NULL	DEFAULT: 600,
Ono:	número del propietario	NOT NULL,	
Sno:	nº de personal en plantilla,		
Bno:	número de sucursal	NOT NULL,	

CLAVE PRIMARIA (Pno),
CLAVE EXTERNA (Sno), REFERENCIAS personal en plantilla, modo de borrado SET NULL, modificación CASCADE,
CLAVE EXTERNA (Ono), REFERENCIAS propietario, modo de borrado NO ACTION, modificación CASCADE,
CLAVE EXTERNA (Bno) REFERENCIAS sucursal, modo de borrado NO ACTION, modificación CASCADE
)

La relación tiene los atributos con los mismos nombres, y tipos de dominios como hemos descrito anteriormente en el DBDL. La clave primaria de la relación es el número propietario, Pno. SQL automáticamente fuerza a un único atributo. Tres claves alternas han sido identificadas con apropiada fuerza referencial. Por ejemplo, el número de personal en plantilla, Son, es una clave alterna referenciando a la relación personal en plantilla. Una norma de borrado ha sido especificada (modo de borrado SET NULL) así que, si el número de personal en plantilla en la relación personal, es borrada, entonces el correspondiente valor o valores para el atributo Son en la relación Propietario_de la_Renta es SET NULL. El número del propietario, Ono, es una clave alterna referenciando a la relación propietario. Las reglas de actualización han sido especificadas (modo de actualización en CASCADE), así que, si el número del propietario en la relación propietario es modificada, entonces el correspondiente valor en el atributo Ono en la relación Propietario_de la_Renta tomaría un nuevo valor (así que, el modo de actualización cascades). En adición, tenemos que jugar a dejar algún valor: por ejemplo, tenemos asignado para dejar un valor la ´f ´ como atributo tipo.

Desencadenamiento (TRIGGER)

Algunos sistemas, así como Oracle proporcionan el desencadenamiento. Un desencadenamiento es una acción asociada con unos eventos que causan un cambio en el contenido de una relación. Existen tres eventos que pueden desencadenar una acción en ORACLE, como son INSERT, UPDATE, o DELETE de tuplas de una relación. El desencadenamiento puede usarse para forzar o suplir la integridad referencial, para forzar las reglas complejas del negocio, y a revisar los cambios de los datos.

El siguiente ejemplo demostrará como podemos usar Oracle para auditar (revisar) algunos cambios del campo Renta de la relación Propietario_de la_Renta que son superiores al 10%**.**

```
CREATE TRIGGER Propietario antes de la actualización
BEFORE UPDATE ON Propietario_de la _Renta
FOR EACH ROW
WHEN (NEW.rent /OLD.rent >1.1)
        BEGIN
           INSERT INTO Propietario_de la_Renta_Auditora
                VALUES (: OLD.pno, :OLD.street, :OLD.area, :OLD.ciudad, :OLD.pcode, :OLD.tipo, :OLD.habitación, :OLD.renta, :OLD.ono, :OLD.son, :OLD.bno)
        END
```

Estos ejemplos crean un dato modificado desencadenado que se aplica a la relación Propietario_de la_Renta. Así pues, estos desencadenamientos ejecutarán antes una transacción de modificado que ha sido cometida en la Base de Datos. El comando entre BEGIN y END son

puntos clave que serán ejecutados por cada actualización del la relación Propietario_de la_Renta que satisfacen la condición WHEN. Estos desencadenamientos requieren la existencia de una relación llamada Propietario_de la_Renta_Auditora para recibir una copia del registro o de los registros que son modificados y satisfacen la condición cuando son examinados.

INGRES Versión 6.4

En algunos sistemas que no cumplen con el nuevo estándar SQL, no hay soporte para una o más de las claves PRIMARY KEY, FOREING KEY, y DEFAULT.

Similarmente, algunos sistemas no soportan los dominios. Por ejemplo, en INGRES versión 6.4, la relación Propietario_de la_Renta se podría crear en SQL de la siguiente manera:

```
CREATE TABLE Propietario_de la_Renta (
        Pno             VARCHAR (5)         NOT NULL,
        Calle           VARCHAR (25)        NOT NULL,
        Area            VARCHAR (15),
        Ciudad          VARCHAR (15)        NOT NULL,
        Pcode           VARCHAR (8),
        Tipo            CHAR (1)            NOT NULL,
        Habitación      SMALINT             NOT NULL,
        Renta           MONEY               NOT NULL,
        Ono             VARCHAR (5)         NOT NULL,
        Son             VARCHAR (5),
        Bno             VARCHAR (3)         NOT NULL);
```

En estos casos, tenemos que forzar la clave, valores no presentados, y dominios con fuerza dentro de la aplicación y el diseño de estos por consiguiente. La excepción a estos es forzar una clave primaria, donde cada uno puede ser implementado a través de índices únicos, como demostraremos posteriormente, o alternativamente, podríamos especificar la estructura de almacenamiento de cualquier relación, como por ejemplo los árboles B.

Indices únicos

Un índice es un mecanismo de acceso rápido, que recupera los datos desde una relación, similar a la indexación de un libro. Un índice único es uno en cada no dos tuplas de una relación que permite tener el mismo valor del índice. El sistema comprueba para cada valor duplicado cuando la indexación es creada (si existen datos alrededor), y cada cierto tiempo es añadido un dato. Algunos RDBMS soportan la creación de los índices. Un índice único puede forzar para usar únicamente la clave primaria y la clave alternativa. Por ejemplo, la afirmación INGRES CREATE TABLE impide la realización de dos registros siendo insertados dentro de la relación Propietario_de la_Renta con el valor de la misma clave primaria. Una vez solucionado esto, podemos crear un índice único en la clave primaria, campo Pno usando la afirmación INGRES SQL de la siguiente manera:

```
CREATE UNIQUE INDEX Propietario_no_indexado ON Propietario_de la_Renta (Pno);
```

Similarmente, podemos crear un índice compuesto, examinando la relación que está compuesta por la clave primaria y consiste en los atributos (Rno, Pno), usando la afirmación INGRES SQL, de la siguiente manera:

```
CREATE UNIQUE INDEX índice_examinado ON examinado (Rno, Pno);
```

Si la tarjeta DBMS no soporta la definición de la clave primaria y clave alterna, los índices únicos estarán creados para proporcionar esta funcionalidad, si es posible.

Diseño de documentos de la Base Relacional

El diseño de la base relacional, estaría completamente documentada junto con la razón para seleccionar el diseño propuesto. En particular, la razón de documentar es para seleccionar un acercamiento donde existen algunas alternativas.

3.2.1.2; La Fuerza de la Empresa para diseñar la Tarjeta DBMS

OBJETIVO: Poder diseñar las tarjetas DBMS según la fuerza de la empresa.

Actualizadas las relaciones deberán ser fuertes por las normas de la empresa gobernando las transacciones que son representadas por la actualización. El diseño de tales fuerzas es otra vez dependiente en la elección de la tarjeta DBMS; algunos sistemas proporcionan más facilidades que otros para definir la fuerza de la empresa. Como en el paso anterior, si el sistema está de acuerdo con el nuevo estándar SQL, algunas fuerzas deben ser fáciles de implementar. Por ejemplo, *DreamHome* debe tener una norma o propiedad que prevenga a un miembro de implementar la plantilla desde el punto de vista general, más que diez propiedades al mismo tiempo. Podríamos diseñar esta fuerza dentro de la afirmación SQL CREATE TABLE para la propiedad Propietario_de la_Renta, usando la siguiente cláusula:

```
CONSTRAINT  personal en plantilla_no_demasiados
        CHECK (NOT EXISTS          (SELECT son
                                    FROM Propietario_de la_Renta
                                    GROUP BY son
                                    HAVING COUNT (*) > 10 ))
```

Alternativamente, un desencadenamiento (trigger) podría usarse para forzar alguna fuerza. Para este ejemplo en concreto, en algunos sistemas podríamos crear el siguiente desencadenamiento para forzar esta integridad:

```
CREATE TRIGGER Personal en plantilla_no_demasiados
ON Propietario_de la_Renta
FOR INSERT, UPDATE
AS IF ((SELECT COUNT (*) FROM Propietario_de la_Renta p, WHERE
p.Son = INSERTED. Son) > 10)
BEGIN
    PRINT “Miembros de personal en plantilla generalmente alrededor de 10
    propietarios”
    ROLLBACK TRANSACTION
END
```

Esto crea un desencadenamiento que es recurrido cuando una tupla es insertada dentro de la relación Propietario_de la_Renta o cuando existe una tupla modificada. Se comprueba que el número de propietarios de los miembros de personal en plantilla no es en general más grande que 10, y si es más grande, expone un mensaje y aborta la transacción.

En algunos sistemas, no habrá soporte para algunas o todas las fuerzas de la empresa y sería necesario diseñar las fuerzas dentro de la aplicación.

Por ejemplo, hay muy poca relación DBMS (si hay alguna) que sería capaz de manejar un tiempo de fuerza, así como a las 5:30 del último día de trabajo de cada año, archivar los registros de todas las propiedades vendidas en ese año y suprimir los registros asociados.

Documentos diseñados para la fuerza de la empresa

El diseño de la fuerza de la empresa debería estar completamente documentado. En particular, las razones de los documentos son para seleccionar un acceso donde existen algunas alternativas.

4.-REPRESENTACIÓN DEL DISEÑO FÍSICO

OBJETIVO: determinar los ficheros de las organizaciones y métodos de acceso que serán usados para almacenar la base relacional; esto es, la manera en que cada tupla estarán contenidas en un almacenamiento secundario.

Uno de los principales objetivos del Diseño de la Base de Datos Física es el almacenamiento de datos de una forma eficiente. Hay un número de factores que debemos usar para medir eficientemente:

- *Introducción de transacción*: esto es un número de transacciones que pueden ser procesadas en un intervalo de tiempo. En algunos sistemas, así como en reservas de líneas aéreas, al introducir una gran transacción es crítico el éxito total del sistema.
- *Tiempo de reacción*: esto es el tiempo transcurrido para la conclusión de una sola transacción. Desde el punto de vista de los usuarios querríamos minimizar el tiempo de reacción tanto como sea posible. De cualquier manera, hay algunos factores que influyen en el tiempo de reacción sobre los que el diseñador no debe tener control. Tal como sistemas cargados o el tiempo de comunicación.
- *Almacenamiento en disco*: este es la cantidad de espacio del disco requerido para el almacenamiento de los ficheros de la Base de Datos. El diseñador debe desear minimizar la cantidad usada de almacenamiento del disco.

De cualquier manera, no hay un factor que sea siempre correcto. Típicamente, el diseñador tiene que negociar un factor de apagado contra otro, para conseguir un balance razonable. Por ejemplo, aumentando la cantidad de datos almacenados, debe disminuir el tiempo de respuesta, o introduciendo una transacción. El diseño inicial de la Base de Datos Física no será considerado como estática, pero será considerada como una estimación del funcionamiento operacional. Una vez que el diseño inicial ha sido implementado, será necesaria la monitorización del sistema y el tono, como resultado del funcionamiento observado y los requerimientos cambiantes.

Algunos DBMS, proporcionan al Administrador de la Base de Datos (DBA) con utilidades a la operación monitora del sistema y sintonizarlo. Veremos que hay algunas estructuras de almacenamiento que son eficientes para cargar el volumen de datos dentro de la Base de Datos, pero ineficiente después de esto. En otras palabras, debemos cambiar para establecer la estructura eficiente del uso del almacenamiento para establecer la Base de Datos y entonces cambiarlo por un uso operacional.

Otra vez, el tipo de organizar los ficheros disponibles son dependientes de la tarjeta DBMS; algunos sistemas proporcionan mas elecciones de estructuras de almacenamiento que otros. Esto es extremadamente importante para que el diseñador de la Base de Datos Física, comprenda totalmente las estructuras de almacenamiento que están disponibles, y además, como la tarjeta de los sistemas usa estas estructuras.

Esto debe requerir que el diseñador conozca como las preguntas del sistema optimizan las funciones. Por ejemplo, debe haber circunstancias donde las preguntas optimizadas no usarían un índice secundario aún si una estuviera disponible. De este modo, aumentando el índice secundario no mejoraría el funcionamiento de la pregunta, y el resultado estaría injustificado.

Algunos sistemas permiten a los usuarios inspeccionar estrategias para ejecutar una pregunta particular o actualizarla, a veces llamado el **Query Execution Plan (QEP)** .

Por ejemplo, DB2 tiene una utilidad explicativa (EXPLAIN), Oracle tiene un plan explicativo (EXPLAIN PLAN), de utilidad diagnóstica, y INGRES tiene un Online QEP (facilidad de criterio). Cuando una pregunta se ejecuta más despacio de lo esperado, esto vale usándolo como una facilidad para determinar la razón de la lentitud, y encontrar una estrategia alternativa que deba mejorar el funcionamiento de la pregunta.

Entendimiento de los recursos del sistema

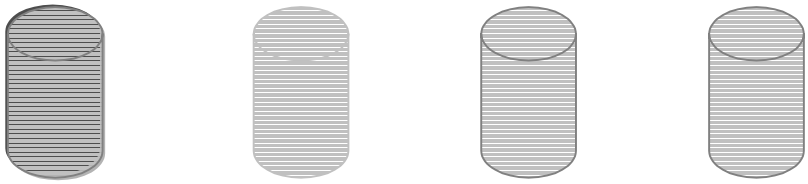
Para mejorar el funcionamiento, del diseño de la Base de Datos Física debe ser consciente de cómo los cuatro componentes básicos del Hardware interactuan y afectan al funcionamiento del sistema:

- *Memoria principal*: El acceso a la memoria principal es significativamente más rápido que el acceso al almacenamiento secundario, a veces diez o hasta ciento de miles de veces más rápida. En general, la memoria principal más disponible al DBMS y a las aplicaciones de la Base de Datos, la aplicación más rápida es la que se ejecutará. De cualquier manera, es sensible siempre para tener un mínimo del 5% de la memoria principal disponible.
  Igualmente es aconsejable no tener nada más que un 10% disponible, de otra manera la memoria principal no está siendo usada de forma óptima. Cuando hay insuficiente memoria para acomodar a todos los procesos, el sistema operativo transfiere las páginas requeridas, el sistema operativo tiene que transferirlo desde el disco. Algunas veces, es necesario intercambiar todos los procesos desde la memoria al disco, y volver otra vez a liberar memoria. Ocurren problemas con la memoria principal cuando se hace una paginación o cuando se llega a ejecutar un intercambio.

- *CPU*: la CPU controla las tareas de recursos de otros sistemas y el proceso de ejecución. El principal objetivo de este componente es prevenir el argumento en cada proceso cuando está esperando la CPU. CPU se encuentra embotellada cuando cada sistema operativo hace demasiadas llamadas a los programas en la CPU.
  Esto es a menudo el resultado de una excesiva paginación o intercambio.

- *DISK I/O*: con algún gran DBMS, hay una cantidad significante del disco I/O introducido en el almacenamiento y recuperación de datos. El uso del disco (disk) I/O tiene una velocidad recomendable. Cuando esta velocidad es excedida I/O se produce un embotellamiento. La forma en que los datos están organizados en disco puede tener un impacto mayor sobre todo el funcionamiento del disco. Es recomendable que el almacenamiento sea uniformemente distribuido a través del almacenamiento en los drivers para reducir la probabilidad de que ocurran problemas.

  La siguiente figura muestra los principios básicos de distribución de los datos a través de los discos:



Sistema Operativo | Ficheros de la Base de Datos Principal | Ficheros Indexados | Recuperación de Ficheros

- El fichero del sistema operativo debería estar separado del fichero de la Base de Datos.
- El fichero de la Base de Datos Principal debería estar separado de los ficheros indexados.
- El fichero de recuperación debería estar separado del resto de la Base de Datos.

- *Red:* cuando la cantidad de tráfico en la red es demasiado grande, o cuando el número de colisiones en la red es muy larga se produce el embotellamiento de la red.

  Cada uno de estos recursos debe afectar a los recursos de otros sistemas. Igualmente un progreso en un recurso debe afectar a un perfeccionamiento en los recursos de otros sistemas.

  Por ejemplo:

  - Conseguir más memorias principales resultaría una menor paginación y un intercambio.
    Esto ayudaría a evitar un embotellamiento en la CPU.
  - Un uso más eficiente de la memoria principal debe resultar un menor disco I/O.

  Con estos objetivos, ahora discutiremos los siguientes pasos:

  - 4.1.- Análisis de transacciones.
  - 4.2.- Seleccionar el fichero en las organizaciones.
  - 4.3.- Añadir índices secundarios.
  - 4.4.- Considerar la introducción de los controles de redundancia.
  - 4.5.- Estimar el espacio del disco requerido.

4.1.- Análisis de transacciones.

OBJETIVO: Comprender la funcionalidad de las transacciones que se ejecutaran en la Base de Datos y analizará las transacciones más importantes.

Una transacción en CA-DB/VAX es una unidad de trabajo sobre la Base de Datos atómica y totalmente recuperable. Todas las acciones de una transacción son totalmente realizadas a deshechas. CA-DB/VAX da al usuario todo el control y flexibilidad que necesita para el tratamiento de transacciones como:

- Transacciones anidadas.
- Savepoint.
- Commits con retención de bloqueos.
- Transacciones con prioridad.

Realizar eficientemente el diseño de la Base de Datos Física, es necesario tener conocimiento de la transacción o preguntas que se ejecutarán en la Base de Datos. Esto incluye ambas informaciones; cualidad y cantidad. Para cada transacción, debemos determinar:

- La frecuencia esperada así como la ejecución de la transacción.
- Las relaciones y el acceso a los atributos por la transacción y el tipo de acceso; esto es Query, Insert, Update y Delete.
  - Por una transacción modificada, los atributos notan que son modificados, cómo estos atributos deben ser candidatos para evitar una estructura de acceso, así como un índice secundario.
- El uso de los atributos en algunos predicados (en SQL, los predicados son las condiciones especificadas en la cláusula WHERE). Comprobar si los predicados implican al modelo que hace juego, búsqueda de cadenas, o poder salvar la clave.
  Estos atributos deben ser candidatos para las estructuras de acceso.

- Para una pregunta, los atributos que son implicados en la unión de dos o más relaciones.
  Otra vez, estos atributos son candidatos para la estructura de acceso.
- La fuerza del tiempo impuesto en las transacciones; por ejemplo, la transacción debe ser completada dentro de un segundo.
  - Los atributos usados en algunos predicados para transacciones críticas deberían tener una propiedad más alta para las estructuras de acceso.

A continuación mostramos un análisis para seleccionar transacciones:

Transacción A

	Día	*hora*	*horas que no ejecuta*
Cima	-	-	-
Camino	-	-	solo ocasionalmente

De relación	*a relación*	*atributos*	*acceso*	*sin acceso de tiempo*
-	sucursal			1
		dirección	R(E)	
		Bno	R	
sucursal	personal de plantilla		I	1
		(todo)		

Transacción B

	Día	*hora*	*horas que no ejecuta*
Cima	Lunes	9-10 AM	2
Camino	Martes	2-4 PM	*2*

De relación	*a relación*	*atributos*	*acceso*	*sin acceso de*
-	sucursal			1
		dirección	R(E)	
		Bno	R	
Sucursal	personal de plantilla			8-20
		Bno	R(E)	
		Sno	R	
		FNombre	R	
		Lnombre	R	
Personal de plantilla	Propietario de la Renta			48-200
		Son	R(E)	
		Pno	R	
		Dirección	R	

Transacción C				
	Día	*hora*	*horas que no ejecuta*	
Cima	Lunes	9-11 AM		4
Camino	Resto de la semana			1
De relación	*a relación*	*atributos*	*acceso*	*sin acceso de*
-	Propietario De la Renta			2
		Pno	R(E)	
		dirección	R	
		Bno	R	
		Son	U	
Propietario de la Renta	personal de plantilla			8-20
		Sno	R	
		Fnombre	R	
		Lnombre	R	
		Bno	R(E)	
Personal de plantilla	Propietario de la Renta	Son	R(E)	6-200

4.2 Seleccionar el fichero en las organizaciones

OBJETIVO: Determinar un fichero eficiente para la organización, para cada relación base.

El objetivo de este paso es seleccionar un fichero óptimo para la organización, para cada relación. Proporcionamos los principios para seleccionar un fichero para la organización basada en los siguientes tipos de ficheros:

- Heap.
- Hash.
- Método de Acceso Secuencial Indexado (ISAM).
- Árboles B^+.

Heap

El fichero de organización Heap es una buena estructura de almacenamiento en las siguientes situaciones:

1) Cuando los datos están empezando a cargar el volumen dentro de la relación. Por ejemplo, para poblar una relación después de ser creada, un lote de registros deben ser insertados dentro de la relación. Si Heap es seleccionado como el fichero inicial

de la organización, debe ser más eficiente para reestructurar el fichero después de que las inserciones hayan sido completadas.
2) Cuando la relación es solo unas cuantas páginas. En este caso, el tiempo para localizar algún registro es corto, aún si la relación completa ha sido seriamente buscada.
3) Cuando cada tupla en la relación ha sido recuperada (en algún orden) cada tiempo la relación es accedida. Por ejemplo, recuperar las direcciones de todos los propietarios de la renta.
4) Cuando la relación tiene una estructura de acceso adicional, así como una clave indexada, el almacenamiento Heap puede usarse para conservar espacio.

Los ficheros Heap son inapropiados cuando solo las tuplas seleccionadas de una relación son accesibles.

Hash

Los ficheros Hash de organización es una buena estructura de almacenamiento cuando las tuplas son recuperadas basadas en un modelo exacto del campo valor.

Por ejemplo, si a la relación del Propietario_de la_Renta le aplicamos Hash en Pno, la recuperación de la tupla con Pno es igual a SG37 es también eficiente. Hash no es una buena estructura de almacenamiento en las siguientes situaciones:

1) Cuando las tuplas son recuperadas basándonos en un modelo patrón del valor del fichero Hash. Por ejemplo, recuperadas todas las propiedades del número del propietario, Pno, empiezan con los caracteres “SG”.
2) Cuando las tuplas son recuperadas basándonos en un valor de rango para el fichero Hash. Por ejemplo, recuperadas todas las propiedades con una renta en el rango 300-500.
3) Cuando las tuplas son recuperadas basándonos en otro fichero Hash. Por ejemplo, si a la relación personal de plantilla le aplicamos Hash en Son, entonces Hashing no puede usarse para buscar una tupla basada en el atributo Lnombre. En este caso, sería necesario realizar un registro lineal para encontrar la tupla, o añadir Lnombre como un índice secundario.
4) Cuando las tuplas son recuperadas en solo parte del fichero Hash. Por ejemplo, si a la relación Propietario_de la_Renta se le aplica Hash en Habitación y Renta, entonces Hashing no puede usarse para buscar una tupla basada solo en el atributo Habitación. Otra vez, sería necesario realizar una búsqueda lineal para encontrar la tupla.

Método de Acceso Secuencial Indexado (ISAM)

El fichero de organización secuencial indexado, es una estructura de almacenamiento más versátil que Hash; esto supone recuperaciones basadas en un modelo de clave exacta, modelo patrón, rango de valores, y parte de la especificación de la clave. De cualquier manera el índice ISAM es estático, creado cuando el fichero es creado. Este modo de funcionamiento de un fichero ISAM se deteriora cuando la relación está actualizada. Actualizar también causa en el fichero ISAM perder el acceso de la clave secuencial. Así que recuperar en orden la clave de acceso llega a ser más lento. Estos dos problemas son vencidos por los ficheros de organización árboles B^+ **.** De cualquier manera, los árboles B^+, para acceder a la indexación pueden ser fácilmente dirigidos porque el índice es estático.

Arboles B^+

El fichero de organización Arboles B^+, es una estructura de almacenamiento más versátil que Hashing. Esto supone recuperaciones basadas en el modelo exacto de clave, modelo patrón, rango de valores, y parte de la especificación de la clave. Los Árboles B^+ son índices dinámicos, es decir, van creciendo cuando la relación crece. Así mismo, el ISAM es distinto, el funcionamiento de un fichero Arbol B^+ no se deteriora cuando la relación es actualizada. Los árboles B^+ también mantienen el orden de la clave de acceso, igualmente cuando el fichero es actualizado. Así mismo recuperar las tuplas en el orden de la clave de acceso es más eficiente que ISAM. Así que, si la relación no es frecuentemente actualizada, la estructura ISAM debe ser más eficiente cuando se tiene un nivel menos de índices que en los árboles B^+, los nodos hojas contienen los registros punteros.

Documentos elegidos para la organización de ficheros

La elección para organizar los ficheros deberían estar documentados completamente junto con las razones de la elección. En particular, las razones de documentar son para seleccionar un acercamiento donde existen algunas alternativas.

4.3 Añadir índices secundarios

OBJETIVO: Para determinar si se añaden índices secundarios mejoraremos el funcionamiento del sistema.

Los índices secundarios proporcionan un mecanismo para especificar una clave adicional para la relación base que puede ser usada para recuperar datos más eficientemente. Por ejemplo, a la relación Propietario_de la_Renta se le debe aplicar Hashing en el número del propietario, Pno, como **índice primario**. De cualquier manera, allí debe ser un acceso frecuente para esta relación basada en el atributo Renta. En este caso, debemos tomar el atributo Renta como un **índice secundario.** Esto es implementado usando SQL de la siguiente forma:

```
CREATE  INDEX Indice_Propietario_de la_Renta ON Propietario_de la_Renta (Renta);
```

Sin embargo, hay una implicación en el mantenimiento y uso del índice secundario, que tiene que ser balanceado contra el funcionamiento mejorado para recuperar datos. Esto anteriormente citado incluye:

- Añadir un registro índice a cada índice secundario cuando un registro es insertado en la relación.
- Actualizando un índice secundario cuando el correspondiente registro en la relación es actualizado.
- El incremento en el espacio de disco necesita para el almacenamiento el índice secundario.
- La posible degradación del funcionamiento durante la optimización de preguntas, como la optimización de las preguntas debe considerar todos los índices secundarios antes de seleccionar una estrategia de ejecución óptima.

A continuación se muestran los siguientes principios para ayudar a seleccionar los índices:

1) En general, indexar la clave primaria de una relación si no es una clave del fichero de organización. Mientras el estándar SQL2 proporciona una cláusula para la especificación de la clave primaria, se debería tener en cuenta que esto no garantiza que la clave primaria será indexada.
2) No indexar las relaciones pequeñas. Esto debe ser más eficiente para buscar la relación en memoria que almacenar una estructura de índices adicionales.

3) Añadir un índice secundario para algún atributo que es fuertemente usado como una clave secundaria (por ejemplo, añadir un índice secundario a la relación Propietario_de la_Renta basado en el atributo Renta).
4) Añadir un índice secundario para una clave externa si es frecuentemente accedida. Por ejemplo, debemos jugar frecuentemente con la relación Propietario_de la_Renta y la relación propietario con el atributo Ono, el número del propietario. Por lo tanto, debe ser más eficiente añadir un índice secundario a la relación Propietario_de la_Renta basada en el atributo Ono.
5) Evitar indexar un atributo o relación que es frecuentemente actualizado.
6) Evitar indexar un atributo si la pregunta recupera una proporción significante de las tuplas en la relación. En este caso, debe ser más eficiente buscar la relación entera que buscar usando un índice.
7) Evitar indexar atributos que consisten en grandes cadenas de caracteres.

Si un número largo de tuplas están siendo insertadas dentro de una relación con uno o más índices, debe ser más eficiente para la caída del primer índice realizar las inserciones, y entonces recrear el índice después. Como una norma de ojear, si la inserción incrementa el tamaño de la relación al menos un 10%, bajar el índice temporalmente.

Documentos elegidos para los índices secundarios

La elección de los índices debería estar completamente documentado, junto con la razón de la elección. En particular, sí hay razones de funcionamiento porque algunos atributos no están indexados, estos deberían también estar documentados.

4.4 Considerar la introducción de los controles de redundancia

OBJETIVO: Para determinar si la introducción de la redundancia es controlada de manera enervante, con las normas de normalización mejoraremos el funcionamiento del sistema.

La normalización es un procedimiento para decidir que atributos permanecen juntos en una relación. Uno de los conceptos básicos de la teoría relacional es que agrupamos los atributos en una relación porque hay una dependencia funcional entre ellos. El resultado de la normalización es un diseño lógico de la Base de Datos que es estructuralmente consistente y que minimiza la redundancia. Así mismo, esto es a veces discutido que un diseño de la Base de Datos normalizada no proporciona un proceso eficiente máximo. Consecuentemente deben haber circunstancias donde sea necesaria para aceptar la pérdida de más de algunos de los beneficios de un diseño lleno de normalización a favor del funcionamiento. Esto debe ser considerado solo cuando esto es estimado que el sistema no será capaz de reunir los requerimientos del funcionamiento. Por ello no seremos partidarios de que la normalización debe ser omitida del diseño lógico de la Base de Datos: normalización fuerza para comprender cada atributo que tiene que ser representado en la Base de Datos. Esto debe ser un factor menos importante que contribuye a todos los sucesos del sistema. En adición, los siguientes factores van a ser considerados:

- La denormalización hace la implementación más completa.
- La denormalización a menudo sacrifica la flexibilidad.
- La denormalización debe dar rapidez para recuperar, pero es lento para actualizar.

Formalmente, el término **denormalización** se refiere a un refinamiento para el esquema relacional, así que el grado de la normalización para una relación modificada es menor que el grado de al menos una relación original. También usaremos el término más lentamente para referirnos a las situaciones donde combinamos dos relaciones dentro de una nueva relación,

donde la nueva relación está todavía normalizada pero contiene más nulos que la relación original. Denormalización es también llamada **uso de refinamiento**.

A continuación mostramos una tabla de contrareferencias de transacciones y relaciones:

Transacción/ Relación	*conservación propietarios*				*lista de propietarios para cada sucursal*				*lista de las vistas con comentarios*			
	I	R	U	D	I	R	U	D	I	R	U	D
Sucursal		X				X				X		
Personal en plantilla		X										
Propietario_de la_ Renta	X	X	X	X		X				X		
Dueño	X	X	X	X		X						
Vista	X	X	X	X						X		
Renta		X	X	X						X		

I= Insert; R= Read; U= Update; D= Delete

Contrareferencias de transacciones y relaciones

Como una norma general de ojear, si el funcionamiento es insatisfactorio y una relación tiene un porcentaje de actualización bajo y un muy alto porcentaje de dudas, la denormalización debe ser una opción viable. Una matriz transacción/relación contrareferencias proporciona y sirve de información para este paso.

La matriz muestra las transacciones que son requeridas y las relaciones a las que accede, como hemos mostrado en la tabla anterior. La matriz resume, de una forma visual el patrón de acceso de las transacciones que ejecuta en la Base de Datos. Puede ser usada para subrayar los posibles candidatos para la denormalización, y para calcular los efectos que estos tendrán en el modelo. Para ser más servible, podríamos indicar el número de accesos sobre algunos intervalos de tiempo (por ejemplo, cada hora, diariamente, y semanalmente) en cada celda. Así mismo, para guardar la tabla simple, no enseñaremos esta información..

En esta sección vamos a considerar los siguientes pasos principales:

- ***4.4.1: Considerar datos derivados.***
- ***4.4.2: Considerar atributos duplicados o juego de relaciones juntas.***

Vamos a considerar como ejemplo la Base de Datos *DreamHome* que hemos venido usando hasta ahora.

Paso 4.4.1: Considerar datos derivados

El valor de los atributos puede encontrar para examinar el valor de otros atributos que son conocidos como **derivados** o **atributos calculados**. Por ejemplo, los indicados a continuación son todos atributos derivados:

- El número de personal en plantilla que trabaja en un sucursal particular.
- El salario total mensual de toda la plantilla.
- El número de propietarios que cada miembro de la plantilla maneja.

A menudo, los atributos derivados no aparecen en el modelo de datos lógico, pero son documentados en el diccionario de datos. Si un atributo derivado es expuesto en el modelo, una elipse salpicada es usada para indicar que este es derivado. El primer paso entonces, es para examinar el modelo de datos lógico y el diccionario de datos, y produce un listado de todos los atributos derivados.

Desde la perspectiva del diseño de una Base de Datos física, si un atributo derivado es almacenado en la Base de Datos o es calculado cada cierto tiempo esto necesita un Trade-off (empleo-cerrado). El diseñador calcularía:

- El coste adicional para almacenar los datos derivados y guardar consistencia con datos operacionales de cada uno de los derivados.
- El coste para calcular cada tiempo que es requerido.

El o ella entonces seleccionaría las opiniones menos costosas para la fuerza del funcionamiento. Para el último ejemplo citado a continuación, necesitaríamos almacenar un atributo adicional en la relación personal en plantilla representando el número de propietario que cada miembro de la plantilla maneja corrientemente. Una relación personal en plantilla simplificada con los nuevos atributos derivados se muestra a continuación.

El almacenamiento adicional mencionado anteriormente para este nuevo atributo derivado no es particularmente significativo. El campo debería ser actualizado cada cierto tiempo por un miembro de la plantilla que esté asignado para.., o designado para manejar un propietario, o el propietario sea borrado de la lista de propietarios disponibles. En cada caso, el atributo, número de propietarios de los miembros apropiados de la plantilla será incrementado o decrementada en 1.

Las siguientes tablas muestran la relación simplificada de *personal en plantilla* con los atributos derivados *Número de propietarios*:

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**Casa**	**6**	**650**	**CO46**	**SA9**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**llano**	**4**	**400**	**CO87**	**SL41**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**llano**	**3**	**350**	**CO40**	**SG14**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**llano**	**3**	**375**	**CO93**	**SG37**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**casa**	**5**	**600**	**CO87**	**SG37**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**llano**	**4**	**450**	**CO93**	**SG14**	**B3**

PERSONAL EN PLANTILLA

Sno	Fnombre	Lnombre	Calle	NIN	Bno	Nº dePropietarios
SL21	**John**	**White**	**19 Taylor St, Cranford, Londres**	**WK442011B**	**B5**	**0**
SG37	**Ann**	**Beech**	**81 George St, Glasgow PA1 2JR**	**WL432514C**	**B3**	**2**
SG14	**David**	**Ford**	**63 Ashby St, Partick, Glasgow G11**	**WL220658D**	**B3**	**2**
SA9	**Mary**	**Howe**	**2 Elm PI, Aberdeen AB2 3SU**	**WM532187D**	**B7**	**1**
SG5	**Susan**	**Brand**	**5 Gt Western Rd, Glasgow G12**	**WK588932E**	**B3**	**0**
SL41	**Julie**	**Lee**	**28 Malvern St, Kilburn NW2**	**WA290573K**	**B5**	**1**

Sería necesario asegurar que estos cambios fueran hechos consecuentemente para mantener la cuenta correcta, y consecuentemente asegurar la integración de la Base de Datos.

Cuando se accede a una pregunta de estos atributos, el valor es inmediatamente disponible y no tiene que ser calculado. De otro modo, si el atributo no es almacenado directamente en la relación personal en plantilla, debe ser calculado cada cierto tiempo que sea requerido. Esto supone un juego de personal en plantilla y la relación Propietario_de la_Renta. Así, si estos tipos de preguntas son frecuentes o es considerada como crítica para el funcionamiento propuesto, debe ser más apropiado el almacenamiento del atributo derivado, antes de que se calcule cada cierto tiempo.

Debe también ser más apropiado para almacenar atributos derivados, cuando el lenguaje pregunte al sistema, no puede fácilmente hacer una copia del algoritmo para calcular el atributo derivado. Por ejemplo, SQL tiene un juego limitado de agregar funciones y no puede fácilmente manejar las preguntas recursivas.

Paso 4.4.2: Considerar atributos duplicados o juego de relaciones juntas

Este paso es para considerar ciertos atributos duplicados o juego de relaciones juntas, para reducir el número de juegos requeridos para el funcionamiento de una pregunta indirectamente, tenemos encontrado un ejemplo implícito de denormalización cuando se trata con los atributos de la *dirección*. Por ejemplo, consideramos la definición de la relación *sucursal*:

Sucursal (Bno, calle, área, ciudad, Pcodigo, nº teléfono, nº fax)

Estrictamente hablando, esta relación no está en 3ª forma normal: Pcódigo, el correo o código cerrado, funcionalmente determina el *Area* y la *Ciudad*. Así mismo, para normalizar la relación, sería necesario para dividir la relación en dos, de la siguiente forma:

Sucursal(Bno, calle, Pcódigo, nº teléfono, nº fax)
Pos-Código(Pcodigo, area, ciudad)

Así mismo, raramente desearemos acceder a la dirección de la sucursal sin los atributos de área y ciudad. Esto significaría que tendríamos que realizar un juego donde nosotros queremos una dirección completa para una sucursal. Como un resultado, estableceremos una 2ª forma normal e implementaremos la relación original sucursal.

Desdichadamente, no hay normas fijas para determinar cuando se denormalizan las relaciones. Discutiremos algunas de las situaciones más comunes para considerar la denormalización. Para información adicional, el lector interesado es referido para Rogers (1989) y Fleming y Von Hall (1989). En particular, consideraremos la denormalización en las siguientes situaciones, específicamente para acelerar la frecuencia o petición crítica:

- Combinando las relaciones una a una (1:1).
- Duplicando los atributos no clave en relaciones de una a muchas (1:M) para reducir los juegos.
- Referenciar las tablas.
- Duplicando los atributos con clave externa en relaciones de una a muchas (1:M) para reducir los juegos.
- Duplicando los atributos en relaciones de muchos a muchos (N:M) para reducir los juegos.
- Introducir grupos de peticiones.
- Creando tablas extraídas.

Combinando las relaciones una a una *(1:1):* Reexaminar las relaciones 1:1 para determinar los efectos para combinar las relaciones dentro de una relación simple. La combinación debería ser solamente considerada para relaciones que son frecuentemente referidas juntas e infrecuentemente referidas separadamente. Considerar, por ejemplo, la relación 1:1 entre Alquiler y Entrevista. La relación Alquiler contiene información sobre

alquileres y potencia al propietario de la renta; la relación entrevista contiene los datos de la entrevista y comentarios hechos por un miembro de la plantilla sobre una renta. A continuación mostramos las relaciones originales de Alquiler y Entrevista:

ALQUILER

Rno	*Fnombre*	*Lnombre*	*Tipo*	*Renta maxima*	*Bno*
CR76	**John**	**Kay**	**llano**	**425**	**B5**
CR56	**Aline**	**Stewart**	**llano**	**350**	**B3**
CR74	**Mike**	**Ritchie**	**casa**	**750**	**B3**
CR62	**Mary**	**Tregear**	**llano**	**600**	**B7**

ENTREVISTA

Rno	*Sno*	*Fecha*	*Comentarios*
CR62	**SA9**	**14-Mayo-1998**	**Falta urgente de propietarios**
CR56	**SG37**	**28-Abril-1998**	**Arriendo corriente finalizado en Junio**

La relación entre Renta y Entrevista es 1:1 la participación es parcial. Entonces si la participación es parcial, debe haber un número significativo de nulos en la relación combinada, dependiendo de la proporción de tuplas involucradas en la participación. Si la relación Renta es grande y la proporción de tuplas involucradas en la participación es pequeña, habrá una significante cantidad de desperdiciadas. La siguiente tabla muestra una relación combinando Alquiler y Entrevista:

ALQUILER

Rno	*Fnombre*	*Lnombre*	*tipo*	*Renta max*	*Bno*	*Sno*	*Fecha*	*Comentario*
CR76	**John**	**Kay**	**llano**	**425**	**B5**	**nulo**	**nulo**	**Nulo**
CR56	**Aline**	**Stewart**	**llano**	**350**	**B3**	**SG37**	**28-04-98**	**Arrendado acabado en junio**
CR74	**Mike**	**Ritchie**	**casa**	**750**	**B3**	**nulo**	**nulo**	**Nulo**
CR62	**Mary**	**Tregear**	**llano**	**600**	**B7**	**SA9**	**14-05-98**	**Falta urgente de propietarios**

Duplicando los atributos no clave en relaciones de una a muchas *(1:M) para reducir los juegos:* Con el propósito específico de reducir o borrar los juegos desde la frecuencia o preguntas críticas, consideramos los beneficios que deben resultar en la duplicación de uno o más atributos no clave de la relación padre en la relación hijo, en una relación 1:M. Por ejemplo, cuando la relación Propietario_de la_Renta es accedida, son muy comunes los nombres de los dueños para acceder al mismo tiempo. Una pregunta típica en SQL sería:

```
SELECT p.*, o.lnombre
FROM Propietario_de la_Renta p, dueño o
WHERE p.ono = o. Ono AND bno = ´B3´;
```

Basado en la relación original Propietario_de la_Renta y Dueño como se muestra en las siguientes tablas:

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**Casa**	**6**	**650**	**CO46**	**SA9**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**llano**	**4**	**400**	**CO87**	**SL41**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**llano**	**3**	**350**	**CO40**	**SG14**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**llano**	**3**	**375**	**CO93**	**SG37**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**casa**	**5**	**600**	**CO87**	**SG37**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**llano**	**4**	**450**	**CO93**	**SG14**	**B3**

DUEÑO

Ono	*Fnombre*	*Lnombre*	*dirección*	*Nº telefono*
CO46	**Joe**	**Keogh**	**2 Fergus Dr, Banchory, Aberdeen AB2 7SX**	**01224-861212**
CO87	**Carol**	**Farrel**	**6 Achray St, Glasgow G32 9DX**	**0141-357-7419**
CO40	**Tina**	**Murphy**	**63 Well St, Shawlands, Glasgow G42**	**0141-943-1728**
CO93	**Tony**	**Shaw**	**12 Park P1, Hillhead, Glasgow G4 0QR**	**0141-225-7025**

Si duplicamos el atributo Lnombre en la relación Propietario_de la_Renta, podemos borrar la relación dueño desde la pregunta, con SQL sería:

SELECT p.*
FROM Propietario_de la_Renta p
WHERE bno = ´B3´;

Basado en el duplicado del atributo Lnombre en la relación Propietario_de la_Renta, como se muestra en la siguiente tabla:

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	Lnombre	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**Casa**	**6**	**650**	**CO46**	**SA9**	**Keog**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**llano**	**4**	**400**	**CO87**	**SL41**	**Farre**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**llano**	**3**	**350**	**CO40**	**SG14**	**Mur phy**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**llano**	**3**	**375**	**CO93**	**SG37**	**Shaw**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**casa**	**5**	**600**	**CO87**	**SG37**	**Farre**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**llano**	**4**	**450**	**CO93**	**SG14**	**Shaw**	**B3**

El beneficio que resulta de estos cambios tiene que ser balanceados contra los problemas que deben surgir. Por ejemplo, si el dato duplicado es cambiado en la relación padre, debe ser actualizada en la relación hijo. Además, para una relación 1:M, deben ser ocurrencias múltiples de cada item de datos en la relación hijo (por ejemplo, el nombre Farre y Shaw ambos aparecen dos veces revisadas en la relación Propietario_de la_Renta). Así esto es necesario para mantener consistencia de copias múltiples. Si la actualización del atributo Lnombre en la relación Dueño y Propietario_de la_Renta no puede ser automática, el potencial para perder la integridad es considerable. Un problema asociado con la duplicación es el tiempo adicional que es requerido automáticamente para mantener la consistencia cada tiempo que es insertado un registro, actualizado o borrado. En nuestro caso, es improbable que el nombre del dueño de una propiedad se cambie, así la duplicación debe ser garantizada.

Otro problema a considerar es el incremento en el espacio de almacenamiento resultante de la duplicación. Otra vez, con el relativamente bajo coste del almacenamiento secundario actualmente, esto no debe ser tanto un problema. Por lo tanto, estos nos justifican una duplicación arbitraria.

Referenciar las tablas: Referenciar tablas, algunas veces llamadas tablas visitadas o lista pico, son un caso especial de relaciones 1:M. Típicamente, una tabla visitada contiene un código y una descripción. Por ejemplo, debemos definir una tabla visitada (padre) para la propiedad tipo y modificar la tabla Propietario_de la_Renta (hijo), como se muestra en la siguiente tabla. Llamamos a la tabla que mira hacia arriba Tipo-propietario para modificar el tipo de atributo de la relación Propietario_de la_Renta:

TIPO-PROPIETARIO

Tipo	*Descripción*
1	**Casa**
2	**Llano**

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	Lnombre	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**Casa**	**6**	**650**	**CO46**	**SA9**	**Keog**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**Llano**	**4**	**400**	**CO87**	**SL41**	**Farre**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**Llano**	**3**	**350**	**CO40**	**SG14**	**Murphy**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**Llano**	**3**	**375**	**CO93**	**SG37**	**Shaw**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**Casa**	**5**	**600**	**CO87**	**SG37**	**Farre**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**Llano**	**4**	**450**	**CO93**	**SG14**	**Shaw**	**B3**

Las ventajas de usar una tabla que mira hacia arriba es:

- Reducción en el tamaño de la relación hijo; el tipo de código ocupados como 1 byte opuestos para 5 bytes como el tipo de descripción.
- Si la descripción puede cambiar (el cual no es la causa en este ejemplo particular), es más fácil cambiarlo una vez en la tabla que mira hacia arriba, como oponerse al cambio muchas veces en el tiempo en la relación hijo.
- La tabla que mira hacia arriba puede ser usada para validar el input del usuario.

Si la tabla que mira hacia arriba es usada con frecuencia o en preguntas críticas, y la descripción es improbable para cambiar, ésta consideración debería dar una duplicación de los atributos *descripción* en la relación hijo, como se muestra en la siguiente tabla:

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	*Descripcion*	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**1**	**6**	**650**	**CO46**	**SA9**	**Casa**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**2**	**4**	**400**	**CO87**	**SL41**	**Llano**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**2**	**3**	**350**	**CO40**	**SG14**	**Llano**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**2**	**3**	**375**	**CO93**	**SG37**	**Llano**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**1**	**5**	**600**	**CO87**	**SG37**	**Casa**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**2**	**4**	**450**	**CO93**	**SG14**	**Llano**	**B3**

La tabla original que mira hacia arriba no es redundante, por ello puede todavía ser usado para validar el input. Así mismo, para duplicar la descripción en la relación hijo, tenemos que eliminar lo necesario para jugar en la relación hijo, para la tabla que mira hacia arriba.

Duplicando los atributos con clave externa en relaciones de una a muchas *(1:M) para reducir los juegos:* otra vez, con el propósito específico de reducir o remover los juegos desde la frecuencia o preguntas críticas, consideremos los beneficios que deben resultar en la duplicación de uno a muchos de los atributos de la clave externa en una relación. Por ejemplo, una pregunta frecuente para *DreamHome* es alistar todos los propietarios, y dueños como una sucursal, usando una pregunta SQL de la siguiente forma (basado en la relación original mostrada en las relaciones Dueño y Propietario_de la_Renta vistas anteriormente):

```
SELECT o.lnombre
FROM propietario_de la_renta p, dueño o
WHERE p.ono = o.ono AND p.bno = ´B3´;
```

En otras palabras, para conseguir la lista de dueños tenemos que usar la relación Propietario_de la_Renta que tiene un número de sucursal requerido, Bno (mostrado en la tabla anterior). Podemos mover lo necesario de estos juegos para duplicar la clave externa Bno en la relación Dueño; esto es, introducir una relación directa entre las relaciones Sucursal y Dueño. En este caso, podemos simplificar la pregunta en SQL de la siguiente forma:

```
SELECT o.lnombre
FROM dueño o
WHERE bno = ´B3´;
```

A continuación mostramos una tabla basada en la nueva relación dueño, es decir, muestra un duplicado de la clave externa Bno en la relación dueño:

DUEÑO

Ono	*Fnombre*	*Lnombre*	*Dirección*	*Bno*	*Nº telefono*
CO46	**Joe**	**Keogh**	**2 Fergus Dr, Banchory, Aberdeen AB2 7SX**	**B7**	**01224-861212**
CO87	**Carol**	**Farrel**	**6 Achray St, Glasgow G32 9DX**	**B5**	**0141-357-7419**
CO40	**Tina**	**Murphy**	**63 Well St, Shawlands, Glasgow G42**	**B3**	**0141-943-1728**
CO93	**Tony**	**Shaw**	**12 Park P1, Hillhead, Glasgow G4 0QR**	**B3**	**0141-225-7025**

Si estos cambios realizados serían necesarios para introducir la fuerza adicional de la clave externa. Si un dueño alquilara propietarios a través de muchas sucursales, no trabajaría. En este caso, sería necesario para modelar una relación muchos a muchos entre sucursal y dueño. Además que solamente la razón de la relación Propietario_de la_Renta tiene el atributo Bno, que esto es posible para que no haya miembros propietarios de personal en plantilla asignados para ello, particularmente al principio cuando el Propietario es tomado primero para la agencia. Si la relación Propietario_de la_Renta no tiene el número de la sucursal, sería necesario para jugar, la relación Propietario_de la_Renta para la relación personal en plantilla basada en el atributo Son para conseguir el número de la sucursal requerido. La pregunta original en SQL llegaría a ser:

```
SELECT o.lnombre
FROM personal en plantilla s, Propietario_de la_Renta p, dueño o
WHERE s.son = p.son AND p.ono = o.ono AND s.bno = ´B3´;
```

Removiendo dos juegos desde la pregunta debe proporcionar la justificación más grande para duplicar la clave externa en la relación dueño.

Duplicando los atributos en relaciones de muchos a muchos *(N:M) para reducir los juegos:* durante el proceso de modelado de datos lógicos, transformamos cada relación M:N en dos relaciones 1:M. Esta transformación introduce una tercera relación intermedia. Ahora, si deseamos producir información desde la relación M:N, tenemos para jugar 3 entidades: las dos entidades originales y la nueva relación intermedia. En algunas circunstancias, debe ser posible reducir el número de relaciones para ser jugadas para atributos duplicados desde una de las entidades originales en la relación intermedia.

Por ejemplo, la relación M:N entre Alquiler y Propietario_de la_Renta ha sido descompuesta para introducir la relación intermedia Entrevista. Considerar el requerimiento que el *DreamHome* vende sucursales y desea ponerse en contacto con propietarios de estos alquileres, pero tienen todavía que hacer un comentario de la propiedad. Así mismo, ventas de la plantilla necesitan solo los atributos de la calle de la propiedad cuando hablan a los alquilados.

En SQL sería:

```
SELECT p.calle, r.*, v.fecha
FROM alquiler r, entrevista v, Propietario_de la_Renta p
WHERE v.pno = p.pno AND r.rno = v.rno AND comentario IS NULL;
```

A continuación se muestran las relaciones originales de Propietario_de la_Renta, alquiler y entrevista:

PROPIETARIO_DE LA_RENTA

Pno	Calle	Area	Ciudad	Pcodigo	Tipo	Habit.	Renta	Ono	Sno	Bno
PA14	**16 Holhead**	**Dee**	**Aberdeen**	**AB75SU**	**Casa**	**6**	**650**	**CO46**	**SA9**	**B7**
PL94	**6 Argyll St**	**Kilbur**	**Londres**	**NW2**	**Llano**	**4**	**400**	**CO87**	**SL41**	**B5**
PG4	**6 Lawrence**	**Partic k**	**Glasgow**	**G11 9QX**	**Llano**	**3**	**350**	**CO40**	**SG14**	**B3**
PG36	**2 Manor Rd**		**Glasgow**	**G32 4QX**	**Llano**	**3**	**375**	**CO93**	**SG37**	**B3**
PG21	**18 Dale Rd**	**Hyndl**	**Glasgow**	**G12**	**Casa**	**5**	**600**	**CO87**	**SG37**	**B3**
PG16	**5 Novar Dr**	**Hybdl**	**Glasgow**	**G12 9AX**	**Llano**	**4**	**450**	**CO93**	**SG14**	**B3**

ALQUILER

Rno	*fnombre*	*lnombre*	*Dirección*	*Nº teléfono*	*tipo*	*Max-renta*	*Bno*
CR76	**John**	**Kay**	**56 High St, Putney, Londres SW1 4EH**	**0171-774-5632**	**llano**	**425**	**B5**
CR56	**Aline**	**Stewart**	**64 Fern Dr, Pollock, Glasgow G42 0BL**	**0141-848-1825**	**llano**	**350**	**B3**
CR74	**Mike**	**Ritchie**	**18 Tain St, Gourock PA1G 1YQ**	**01475-392178**	**casa**	**750**	**B3**
CR62	**Mary**	**Tregear**	**5 Tarbot Rd, Kildary, Aberdeen AB9 3ST**	**01224-196720**	**llano**	**600**	**B7**

ENTREVISTA

Rno	*Pno*	*fecha*	*Comentario*
CR56	**PA14**	**24-mayo-98**	**También pequeño**
CR76	**PG4**	**20-abril-98**	**También remoto**
CR56	**PG4**	**26-mayo-98**	
CR62	**PA14**	**14-mayo-98**	**Sin habitación comedor**
CR56	**PG36**	**28-abril-98**	

Si duplicamos el atributo *calle* en la relación intermedia entrevista, podemos remover la relación Propietario_de la_Renta desde la pregunta, dando la siguiente pregunta en SQL:

```
SELECT r.*, v.calle, v.fecha
FROM alquiler r, entrevista v
WHERE r.rno = v.rno AND comentario IS NULL;
```

A continuación se muestran la duplicación del atributo Calle en la relación entrevista:

ALQUILER

Rno	*fnombre*	*lnombre*	*Dirección*	*Nº teléfono*	*tipo*	*Max-renta*	*Bno*
CR76	**John**	**Kay**	**56 High St, Putney, Londres SW1 4EH**	**0171-774-5632**	**llano**	**425**	**B5**
CR56	**Aline**	**Stewart**	**64 Fern Dr, Pollock, Glasgow G42 0BL**	**0141-848-1825**	**llano**	**350**	**B3**
CR74	**Mike**	**Ritchie**	**18 Tain St, Gourock PA1G 1YQ**	**01475-392178**	**casa**	**750**	**B3**
CR62	**Mary**	**Tregear**	**5 Tarbot Rd, Kildary, Aberdeen AB9 3ST**	**01224-196720**	**llano**	**600**	**B7**

ENTREVISTA

Rno	*Pno*	*fecha*	*Comentario*
CR56	**PA14**	**24-mayo-98**	**También pequeño**
CR76	**PG4**	**20-abril-98**	**También remoto**
CR56	**PG4**	**26-mayo-98**	
CR62	**PA14**	**14-mayo-98**	**Sin habitación comedor**
CR56	**PG36**	**28-abril-98**	

*Introducir grupos de peticiones: R*epitiendo grupos eliminaremos desde el modelo de datos lógico, como un resultado de los requerimientos de que todas las entidades están en primera forma normal. La repetición de los grupos es separada en nuevas relaciones, formando una relación 1:M con la relación original (padre). Ocasionalmente reintroducir grupos repetidos es una efectiva manera para mejorar el funcionamiento del sistema. Por ejemplo, cada oficina sucursal *Dreamhome* debe de disponer de un número de coches en una compañía, que pueden ser usados por el personal de la plantilla cuando visitan a los propietarios. No todas las oficinas tienen una compañía de coches; la primera tiene un máximo de 4 coches. Típicamente, la compañía de datos de coches debe ser externa para formar una nueva relación hijo, con la clave primaria como el número de registro del coche y una clave externa como el número de sucursal Bno, como muestra la siguiente tabla, las relaciones originales de sucursal y coches de la sucursal:

SUCURSAL

Bno	*Calle*	*Area*	*Ciudad*	*Pcodigo*	*nº tlno*	*nº fax*
B5	**22 Deer Rd**	**Sidcup**	**Londres**	**SW1 4EH**	**0171-886-1212**	**0171-886-1214**
B7	**16 Argyll St**	**Dyce**	**Aberdeen**	**AB2 3SU**	**01224-67125**	**01224-67111**
B3	**163 Main St**	**Partick**	**Glasgow**	**G11 9QX**	**0141-339-2178**	**0141-339-4439**
B4	**32 Manes Rd**	**Leigh**	**Bristol**	**BS99 INZ**	**0117-916-1170**	**0117-776-1114**
B2	**56Clover**		**Londres**	**NW10 6EU**	**0181-963-1030**	**0181-453-7992**

COCHES-SUCURSAL

Bno	*Nº de registro*
B5	**M109 ABG**
B5	**M670 BFT**
B5	**N64 SAP**
B7	**M536 XRT**
B7	**N90 BDC**

Si el acceso a esta información es importante o una pregunta de frecuencia debe ser más eficiente para combinar las relaciones y almacenar los detalles de los coches en la relación original sucursal, con una columna para cada coche, la siguiente tabla muestra la relación sucursal revisada con grupos repetidos:

SUCURSAL

Bno	*Calle*	*área*	*ciudad*	*Pcódigo*	*Nº tlno*	*Coche1*	*Coche2*	*Coche 3*	*Coche 4*	*Nº fax*
B5	**22 Deer Rd**	**Sidcup**	**londres**	**SW1 4EH**	**0171-886-1212**	**M109 ABG**	**M670 BFT**	**N64 SAB**	**nulo**	**0171-886-1214**
B7	**16 Argyll St**	**Dyce**	**Aberdeen**	**AB2 3SU**	**01224-67125**	**M536 XRT**	**N90 BDC**	**nulo**	**nulo**	**01224-67111**
B3	**163 Main St**	**Partick**	**Glasgow**	**G11 9QX**	**0141-339-2178**	**…**	**…**	**…**	**…**	**0141-339-4439**
B4	**32 Manse Rd**	**Leigh**	**Bristol**	**BS99 INZ**	**0117-916-1170**	**…**	**…**	**…**	**…**	**0117-776-1114**
B2	**56 Clover Dr**		**Londres**	**NW10 6EU**	**0181-963-1030**	**…**	**…**	**…**	**…**	**0181-453-7992**

En general, estos tipos de denormalización se deberían considerar solo en las siguientes circunstancias:

- El número absoluto de ítems en los grupos repetidos es conocido (en este ejemplo, hay una máximo de 4 coches).
- El número es estático y no cambiará sobre el tiempo (el número de coches es fijado por la compañía).
- El número no es muy largo típicamente no más grande que doce, aunque esto no es tan importante como las dos condiciones anteriores.

A veces, debe ser solo el más reciente o corriente valor en un grupo repetido, o justo el hecho de que hay un grupo repetido, que es necesario más frecuentemente. En el ejemplo de arriba, necesitamos almacenar solo el hecho de que una oficina tiene una compañía de coches. En este caso, podríamos considerar almacenar solo una columna extra en la relación sucursal representando esta información.

Creando tablas extraídas: Debe haber situaciones donde los informes tienen que ejecutarse para ser la cumbre durante el día. Estos informes de datos de accesos derivados y realizar multirelaciones de juegos en el mismo set de las relaciones base. Así mismo, el informe del dato está basado en que debe ser relativamente estático o en algunos casos, esto no tiene que ser corriente (esto es, si el dato fuera unas pocas horas, el informe sería perfectamente aceptable). En estos casos, sería posible crear uno simple, más alto denormalizar la tabla extraída basada en las relaciones requeridas para el informe, y permite a los usuarios acceder a la tabla extraída directamente en lugar de a la relación base. La técnica más común para producir la tabla extraída es crear y poblar las tablas en una colección ejecutada cuando el sistema está ligeramente cargado.

Introducción de documentos redundantes

La introducción de la redundancia debería estar completamente documentada, junto con las que introduce. En particular, la razón de los documentos para seleccionar un acceso aproximado donde existen muchas alternativas. Actualizar el modelo de datos lógicos para reflejar algunos cambios hechos como un resultado de la denormalización.

4.5.- Estimar el espacio del disco requerido

OBJETIVO: Para estimar la cantidad del espacio del disco que será necesaria para la Base de Datos.

Debe ser un requisito que la implementación de la Base de Datos Física pueda ser manejada por la configuración del Hardware corriente. Si este no es el caso, el diseñador todavía tiene que estimar la cantidad del espacio en disco que es necesaria para almacena la Base de Datos. En el suceso que los nuevos Hardware tienen que ser obtenidos. El objetivo de estos pasos es para estimar la cantidad de espacio del disco que es necesario para el soporte de la implementación de la Base de Datos en almacenamiento secundario. Como con los pasos previos, estimar el disco usado es muy dependiente de la tarjeta DBMS y el Hardware usado para soportar la Base de Datos. En general, el estimar está basado en la cantidad de cada tupla y el número de tuplas en la relación. La última estimación debería ser un número máximo, pero debería también valer en consideración a como la relación aumentará, y modificando el resultado de la cantidad de disco por este factor de desarrollo para determinar la cantidad de potencial de la Base de Datos en el futuro.

Ilustramos este proceso usando la organización de ficheros anteriormente vistos, utilizando el INGRES DBMS. Las fórmulas están basadas en recientes relaciones pobladas, antes de que los datos hayan sido borrados o añadidos. En general, el cómputo para la cantidad de espacio requerido para implicar una relación multiplicando el número de tuplas por el tamaño de una tupla y añadirlo al tamaño para algunos índices requeridos. El tamaño de una tupla es una suma del tamaño del atributo más algunos de arriba introducidos por el sistema. El tamaño de un atributo es determinar dividir por su tamaño y tipo. En INGRES, el tamaño de una tupla es calculado como el número total de bytes por tupla, incluyendo 2 byte de arriba para cadenas de caracteres de longitud variable y un adicional byte para campos nulos.

Heap

Una página INGRES es 2048 bytes, de los cuales 40 bytes son usados como una página de encabezado y los sobrantes 2008 bytes están disponibles para almacenar los datos del usuario. El espacio total requerido para un fichero Heap puede ser calculado de la siguiente forma:

Fila_por_página = 2008/ (ancho_fila + 2); redondeando los enteros hacia abajo.
Total_páginas_Heap = Nº fila/filas_por_página; redondeando los enteros hacia abajo.

Donde la anchura de la fila es el tamaño de la tupla. Por ejemplo, para determinar el tamaño de la relación Propietario_de la_Renta con 10000 registros almacenados como un fichero Heap, tenemos:

Fila_por_página = 2008/(111+2) = 17
Total_página_Heap = 10000/17 = 589

Por tanto, la relación Propietario_de la_Renta necesitaría 589 páginas INGRES para ser almacenada como un fichero Heap. En un sistema con un tamaño de bloque de 512 bytes, la relación Propietario_de la_Renta necesita 589 * 2048 / 512 = 2356 bloques de disco.

Hashing

La fórmula para un fichero Hash es la misma que para un fichero Heap. Además, los números de páginas deben ser ajustados para tener en cuenta el llenado del factor, el cuál es el porcentaje de una página que será usada antes de que la página se considere llena. En INGRES, el factor de llenado no presentado para Hash es 50%, a menos que el ancho del registro sea mayor que 1000 bytes, en cuyo caso el factor de llenado es 100%. La randomización permite

acceder a los datos con una única operación de E/S. A partir del valor de la columna Hash este algoritmo obtiene la página donde se encuentra almacenada dicha fila. El cálculo se hace de la siguiente manera:

Fila_por_página = (factor de llenado * 2008) / (ancho de la fila + 2) redondeo hacia abajo
Total_páginas_Hash = número fila / fila por página * (1 / factor de llenado) redondeo hacia arriba.

Por ejemplo, para determinar el tamaño de la relación Propietario_de la_Renta con 10000 registros almacenados como un fichero Hash con un 50% de factor de llenado, tenemos:

Fila_por_página = 0.5 * 2008 / (111 + 2) = 8
Total_páginas_Hash = (10000 / 8) * 2 = 2500

Por lo tanto, la relación Propietario_de la_Renta necesitaría 2500 páginas INGRES para ser almacenadas como un fichero Hash con una 50% de factor de llenado. En un sistema con un tamaño de bloque de 512 bytes, la relación Propietario_de la_Renta necesitaría:
2500 * 2048 / 512 = 10000 bloques de disco.

ISAM

Como un fichero Hash, un fichero ISAM también tiene que tener en cuenta el factor de llenado, pero en adición, se tiene que hacer una concesión para la suma del espacio del índice ISAM que necesitará. En INGRES, el factor de llenado presentado por ISAM es 80%. El cálculo para la cantidad de la relación es:

Fila_por_página = (factor de llenado * 2008) / (ancho_fila + 2); redondeo libre hacia abajo
= 2008 – (fila_por_página * (ancho fila + 2))
si libre > ((2048 – (factor de llenado * 2048)) y ancho fila <= libre
fila_por_página = nº fila / fila_por_página + 1 máximo valor 512
total_datos_página = nºfila / fila_por_página redondeo hacia arriba

El cálculo para el tamaño del índice es:

Clave_por_página = 2008 / (clave_ancho + 2); redondeo hacia abajo
Total_índices_página = total_datos_página / clave_por_página + 1; redondeo hacia arriba

Donde ancho_clave es el tamaño de la clave por columnas. El espacio total requerido por un fichero ISAM es entonces:

Fila_por_páginas	= 0.8 * 2008 / (111 + 2)	= 15
Total_dato_página	= 10000 / 15	= 667
Clave_por_página	= 2008 / (5 + 2)	= 286
Total_índice_página	= 770 / 286 +1	= 4
Total_ISAM_página	= 667 + 4	= 671

Por lo tanto, la relación Propietario_de la_Renta necesitaría 671 páginas INGRES para ser almacenadas como un fichero ISAM, con un índice en el atributo Pno usando un 80% del factor de llenado. En un sistema con un tamaño de bloque de 512 bytes, la relación Propietario_de la_Renta necesitaría 671 * 2048 / 512 = 2648 bloques de disco.

Árboles B^+

Un árbol B^+ incluye páginas índice, páginas intermedias *sprig* cuya finalidad es hojear páginas, hojear páginas cuya utilidad es para datos de páginas. En adición, datos, índices, y páginas hojas pueden usar diferentes factores de llenado. En INGRES, los factores de llenado presentados son 80% para datos y páginas índices, y 70% para páginas hojas. Este método de acceso se usa para aumentar el rendimiento de largas búsquedas. CA-DB/VAX utiliza una versión mejorada de índices B^+ en la cual se mantiene la información del índice separada de los datos. Este método incrementa el rendimiento al eliminar la necesidad de leer datos innecesarios. CA-DB/VAX puede determinar si una fila contiene los valores buscados leyendo solamente el índice. El cálculo para el tamaño de la relación es:

```
Fila_por_página = (dato de llenado * 2010) / (ancho_fila + 2); redondeo libre hacia abajo
        libre = ((2008 – (fila_por_página * (ancho_fila + 2))
        si libre > ((2048 – (dato de llenado * 2048)) y ancho fila <=  libre
          fila_por_página = fila_por_página + 1          máximo valor 512
clave_máxima          = (1964 / (ancho_clave + 6)) – 2;        redondeo hacia abajo
clave_por_hoja = máximo_clave * hoja_llenado;          redondeo hacia abajo, valor mínimo 2
clave_por_índice = máximo clave * índice_llenado;      redondeo hacia abajo, valor mínimo 2

nº_página_hoja = (nº filas / clave_por_hoja) + 1;          redondeo hacia arriba
nº_datos_página = nº_hoja_página * (clave_por_hoja / fila_por_página) +
        MODULO (nº_fila / clave_por_hoja) / fila_por_página;          ambas          divisiones
redondeadas hacia arriba

Nº páginas sprig = 0              si nº_páginas_hoja <= clave_por_índice
Nº páginas sprig = (nº_página_hoja / clave_por_índice);redondeo hacia arriba
Nº páginas índice = 0
                IF (nº páginas sprig > clave_por_índice) THEN
                        X = nº páginas sprig
                DO
                        X = X / clave_por_índice
                        Nº páginas índice = nº páginas índice + X
                WHILE (X > clave_por_índice)
```

El total de espacio necesitado por un fichero árbol B^+ es entonces:

> Total_página_arbol B^+ = nº_dato_página + nº_páginas_hoja + nº páginas sprig + nº páginas índice + 2

Por ejemplo, para determinar el tamaño de la relación Propietario_de la_Renta con 10000 registros almacenados como un fichero indexado árbol B^+ en el atributo Pno usando los factores de llenado, tenemos:

```
Fila_por_páginas      = 0.8 * 2010 / (111 + 2) = 15
máximo_clave          = 1964 / (5 + 6) - 2     = 176
Clave_por_hoja        = 176 * 0.7              = 123
clave_índice          = 176 * 0.8              = 140
nº página hoja        = (10000 / 123) + 1      = 83
nº datos página       = 83 * (123 / 15) + 3    =750
nº páginas sprig= 0
nº páginas índice     = 0
total páginas B+= 750 + 83 + 0 + 0 + 2 = 835
```

Así mismo, la relación Propietario_de la_Renta necesitaría 835 páginas INGRES para ser almacenado como un fichero Árbol B^+, con un índice en el atributo Pno usando los factores de llenado presentados. En un sistema con un tamaño de bloque de 512 bytes, la relación Propietario_de la_Renta necesitaría 835 * 2048 / 512 = 3340 bloques de disco.
A continuación se muestra una tabla, comparando el espacio necesario para la relación Propietario_de la_Renta:

	10.000	20.000	30.000	40.000	50.000
Heap	2.356	4.708	7.060	9.412	11.768
Hash	10.000	20.000	30.000	40.000	50.000
ISAM	2.684	5.360	8.032	10.712	13.388
Arbol B^+	3.340	6.592	9.844	13.100	16.352

Esta tabla muestra el espacio de disco requerido (en 512 byte de bloque) para la relación Propietario_de la_Renta como el número de registros en el fichero incrementan.

5.- DISEÑO DE LOS MECANISMOS DE SEGURIDAD.

OBJETIVO: para diseñar las reglas de seguridad para la Base de Datos como específicas para el usuario.

Una Base de Datos representa un recurso corporativo esencial, y así la seguridad de estos recursos es extremadamente importante. Deberían ser específicos los requisitos de seguridad documentados durante el diseño de la Base de Datos Lógica. El objetivo de este paso es decidir como estas reglas de seguridad serán realizadas. Algunos sistemas ofrecen diferentes facilidades de seguridad que otros. Otra vez, el diseñador de la Base de Datos debe ser consciente de las facilidades ofrecidas por la tarjeta DBMS.

Las actividades en este paso son:

- 5.1: diseño según la perspectiva del usuario.
- 5.2: diseño de normas de acceso.

5.1: Diseño según la perspectiva del usuario

OBJETIVO: para diseñar las perspectivas del usuario.

La primera fase de la Metodología del diseño de la Base de Datos, implica la producción del modelo de datos conceptual para cada perspectiva del usuario. En la segunda fase de la Metodología del diseño de la Base de Datos, estos modelos conceptuales están refinados en el modelo de datos lógico, los cuales estaban unidos subsecuentemente en un modelo de datos lógico global. El objetivo de este paso es diseñar las perspectivas del usuario, basado en el modelo de datos lógico local. En una sola posición DBMS en un computador personal, las perspectivas son usualmente una conveniencia definida para simplificar la solicitud de la Base de Datos. De cualquier manera, en un multi-usuario DBMS, las perspectivas juegan un papel central en la definición de la estructura de la Base de Datos y en el cumplimiento de la seguridad. La siguiente tabla nos muestra un listado de la vista de personal en plantilla 3:

Son	*fnombre*	*lnombre*	*dirección*	*nº tlfno*	*posición*	*sexo*
SG37	Ann	Beech	81 George St, Glasgow PA12JR	0141-848-3345	ATS	M
SG14	David	Ford	63 Ashby St, Partick,Glasgow G11	0141-339-2177	Suplente	H
SG5	Susan	Brand	5 Gt Western Rd, Glasgow G12	0141-334-2001	Manager	M

Normalmente, las perspectivas son creadas usando SQL. Por ejemplo, creamos una vista para personal en plantilla a la sucursal B3 que excluye la información del salario, así que solo el Manager a esa sucursal puede acceder a los detalles del salario para quien trabaja en su oficina. La afirmación en SQL en este caso sería:

```
CREATE VIEW personal en plantilla 3
AS SELECT son, fnombre, lnombre, dirección, nº tlfno, posición, sexo
        FROM personal en plantilla
WHERE bno = ´B3´;
```

Esto crea una vista llamada Personal en Plantilla 3, con los mismos atributos como la relación personal en plantilla, pero excluyendo los atributos: datos de nacimiento, salario, DNI, y Bno. Si alistamos esta vista vemos los datos en la tabla anterior.

Para asegurar que solo el Manager de la sucursal puede ver el atributo salario, personal en plantilla no debería dar acceso a la relación base personal en plantilla. En cambio, deberían dar permiso de acceso a la vista personal en plantilla3, ahí rechazamos sus accesos para los datos de salario.

5.2: Diseño de normas de acceso

OBJETIVO: para diseñar las normas de acceso a las relaciones base y vistas de usuario.

Una forma para proporcionar seguridad es usar el control de acceso facilitado por SQL. Típicamente, los usuarios no deberían dar un acceso directo para las relaciones base. En cambio, debería dar acceso a las relaciones base a través de los diseños de las vistas de usuario en el paso anterior. Esto proporciona un largo grado de datos independientes y aislar al usuario desde los cambios en la estructura de la Base de Datos. Brevemente revisaremos el mecanismo de control de acceso de SQL2.

A casa usuario de la Base de Datos es asignado un identificador de autorización por el DBA; usualmente, el identificador tiene un password asignado, por razones obvias de seguridad. Cada afirmación SQL que es ejecutado por el DBMS es realizada en nombre de un usuario específico. El identificador de autorización es usado para determinar cuáles de los objetivos de la Base de Datos deben hacer referencia los usuarios, y qué operaciones deben ser realizadas en aquellos objetivos. Cada objetivo que es creado en SQL tiene un dueño. El dueño es identificado por el identificador de autorización. El dueño es la única persona que debe conocer la existencia del objetivo y, consecuentemente, realizar algunas operaciones en el objetivo.

Privilegios son las acciones que a un usuario se le permite para cumplir en una relación base o vista. Por ejemplo, SELECT es el privilegio para recuperar datos desde una relación. Cuando un usuario crea una relación usando el SQL la afirmación es: CREATE TABLE, el o ella automáticamente llega a ser dueño de la relación y recibe privilegios completos para la relación. Otros usuarios inicialmente no tienen privilegios en la reciente relación creada. Para darles acceso a la relación, los dueños deben explícitamente concederles los privilegios necesarios usando la afirmación GRANT. Una cláusula WITH GRANT OPTION puede ser especificada con la afirmación GRANT para permitir recibir a los usuarios y pasar los privilegios a otros usuarios. Los privilegios pueden ser recuperados usando la afirmación REVOKE.

Cuando un usuario crea una vista con la afirmación CREATE VIEW, el o ella automáticamente son los dueños de la vista, pero no reciben necesariamente los privilegios completos en la vista. Para crear la vista un usuario debe tener el privilegio SELECT para todas las relaciones que hacen la vista. Así mismo, el dueño solo conseguirá otros privilegios si el o ella contienen esos privilegios para cada relación en la vista.

Por ejemplo, para permitir al usuario MANAGER recuperar filas desde la relación personal en plantilla e insertar, actualizar, y borrar datos desde la relación personal en plantilla, usaremos las siguientes afirmaciones en SQL:

```
GRANT ALL PRIVILEGES
ON personal en plantilla
TO Manager WITH GRANT OPTION;
```

En este caso, MANAGER será capaz de referenciar la relación y todos los atributos en alguna relación que el o ella crean subsecuentemente. Especificamos la cláusula WITH GRANT OPTION así que MANAGER puede pasar esos privilegios a otros usuarios que el o ella ven

adecuados. Como otro ejemplo, podríamos dar a los usuarios el identificador de autorización ADMIN el privilegio SELECT en la relación personal en plantilla usando la siguiente afirmación en SQL:

```
GRANT SELECT
ON personal en plantilla
TO admin;
```

Tiene que omitir la palabra clave con WITH GRANT OPTION así que ADMIN no será capaz de pasar este privilegio a otros usuarios.

Documentos diseñados para vistas de usuario y medidas de seguridad:

El diseño de la vista de usuario individual y los mecanismos de seguridad asociados deberían estar completamente documentados. Si el Diseño Físico afecta al modelo de datos lógico local, estos diagramas deberían también ser actualizados.

6.- MONITORIZACIÓN Y SINTONIZACIÓN DEL SISTEMA OPERACIONAL.

OBJETIVO: Para la monitorización del sistema operacional y mejorar el funcionamiento del sistema para corregir las decisiones del diseño inapropiado o reflejar los requisitos cambiantes.

Como mencionamos anteriormente, el diseño de la Base de Datos Física inicial no debería ser considerada como estática, pero debería ser considerada como una estimación del funcionamiento operacional. Una vez que ha sido implementado el diseño inicial, es necesario para la monitorización del sistema y sintonización como un resultado del funcionamiento observado y requisitos de cambio. Algunos DBMS proporcionan el DBA con utilidades para monitor de la operación del sistema y sintonización.

Hay muchos beneficios para ganar desde la sintonización de la Base de Datos:

- Puede evitar el conseguir un hardware adicional.
- Sería posible para la configuración del hardware con una extensión baja. Esto resulta menos y más barato, hardware y consecuentemente un mantenimiento menos caro.
- Un buen sistema de sintonización produce tiempos de respuesta más rápidos y mejores, a través de los cuales se hacen turnos de usuarios, y por lo tanto la organización es más productiva.
- Mejorar los tiempos de respuesta puede mejorar la moral del personal de la plantilla.
- Mejorar los tiempos de respuesta puede aumentar la satisfacción de los clientes.

Estos dos últimos beneficios son más intangibles que otros. Así, podemos ciertamente afirmar que respuestas lentas desmoralizan la plantilla de personal y potencialmente pierden clientes.

La sintonización es una actividad que no es nunca completa. A través de la fila del sistema, sería necesario para el funcionamiento del monitor, particularmente para facturar los cambios en el medio ambiente y los requisitos de usuario. De cualquier manera, hacer un cambio a un área de un sistema operacional para proporcional el funcionamiento debe tener un efecto adverso en otra área. Por ejemplo, añadir un índice para una relación debe proporcional el funcionamiento de una aplicación, pero debe afectar adversamente a otra, quizá otra aplicación más importante. Por lo tanto, con cuidado deben tomarse las decisiones para hacer los cambios a un sistema operacional. Si es posible, controlar cualquier cambio en una prueba de Base de Datos, o alternativamente, cuando el sistema no es usado totalmente (por ejemplo, fuera de horas de trabajo).

Existen los siguientes monitores de rendimiento:

- Monitor de E/S, ofrece una interfaz similar al monitor de VMS. Realiza la monitorización de E/S de la Base de Datos por tablespace, fichero o disco.
- Monitor de esperas por transacción, el cual muestra qué recurso espera, por quién y por cuánto tiempo. Dispone de una alarma en caso de alcanzarse el límite, y terminación automática al llegar a éste.

7.- ELEMENTOS DE DISEÑO FÍSICO

7.1.-Índice compuesto relacional

- Estructurado físicamente en formato Árbol B.
- Utiliza técnicas de compresión de alto rendimiento.
- El puntero a datos no utiliza direcciones físicas.
- Los punteros referentes a una misma clave se almacenan de forma contigua.
- Diferenciación de tareas de reorganización en precisas y deseables, con distintas prioridades para cada uno de ambos tipos.
- Reutilización de espacio.

7.2.- Área de datos

- Posibilidad de compresión de datos.
- Utilización de diversos métodos para control de espacio disponible, según las características y tamaño del espacio a considerar.
- Básicamente para bloques completos se indica el ámbito del intervalo y para bloques parcialmente utilizables mapas de bits.
- Posibilidad de extender el área de datos sin cargar los mismos.
- Posibilidad de almacenar en una misma área una o varias tablas.
- Cuatro tipos de gestión de espacio por área (fichero físico) en altas (también en modificaciones cuando ello implica cambio de longitud de los datos). Es importante tener en cuenta que esta característica puede variarse cuando se considere deseable sin que ello implique necesidad inmediata de reorganización.

1) *No hay reutilización de espacio*. Conveniente en caso de datos históricos u oficiales en que las bajas constituyen un proceso raramente ejecutado.
2) *Reutilización básica de espacio*. Los datos se añaden a un bloque disponible en aquel momento en memoria para la misma área (es el método que permite mayor rendimiento cuando la actualización de los datos se efectúa básicam3nte on-line). Si ningún bloque de memoria cumple los requisitos necesarios se utiliza un bloque vacío, si existe. Si no hay ningún bloque vacío se utiliza un bloque parcialmente lleno priorizando para la elección
    - espacio disponible en el bloque,
    - valor de la clave nativa.
3) *Wrap around*. Los datos se añaden secuencialmente al área hasta que ésta llega al final. A partir de ese momento el puntero de espacio disponible se traslada al principio del área.
4) *Clustering*. En un mismo bloque de datos se almacena información con un identificativo común. Dicho identificativo debe ser una parte de orden superior de la clave nativa de los datos.

La **clave nativa** es aquella que constituye el orden natural de los datos. Si se efectúa una reorganización marcará el orden de los datos.

Características

- Varias tablas pueden compartir la misma clave.
- En claves compartidas la longitud de cada una de ellas puede diferir del resto.
- Pueden indexarse o no a voluntad los datos a los que corresponderían valores de clave con espacios o ceros binarios.

8.- ORACLE 8

8.1.- Introducción al servidor Oracle

Este capítulo nos proporciona una visión de conjunto del Servidor Oracle. Los temas incluidos en este capítulo son:

8.1.1 Bases de Datos y dirección de la información.
8.1.2 Estructura de la Base de Datos y espacio de la dirección.
8.1.3 Estructura de la Memoria y Procesado.
8.1.3 Concurrencia de los datos y consistencia.
8.1.4 Procesamiento distribuido y Bases de datos distribuidas.
8.1.5 Comenzar y acabar las operaciones.
8.1.6 Seguridad de la Base de Datos.
8.1.7 Bases de Datos de reserva y recuperación.
8.1.8 El modelo relacional para la dirección de la Base de Datos.
8.1.9 Acceso a datos.

8.1.1 Bases de Datos y dirección de la información

Un servidor de Bases de Datos es la clave para solucionar el problema de la dirección de la información. En general, un servidor debe dirigir de forma segura una gran cantidad de datos en un entorno de multiuso. Así que muchos usuarios pueden acceder concurrentemente a los mismos datos. Todo esto debe ser habitual mientras se entrega alta prestación. Un servidor de la Base de Datos debe además prevenir accesos inautorizados y proporcionar soluciones eficientes para recuperar la pérdida.

El servidor Oracle proporciona eficientes y efectivas soluciones con las siguientes características:

1) Cliente/servidor (Procesamiento distribuido); para tomar todas las ventajas que el sistema computador ofrece o network, Oracle permite el procesamiento, para ser dividido entre el servidor de la Base de Datos y el cliente de la aplicación de programas. La ejecución del computador del sistema de dirección de la Base de Datos maneja todo, responsabilizándose del servidor de la Base de Datos, mientras que la ejecución de la estación de trabajo (workstations) concentra la aplicación de Bases de Datos en la interpretación y visualización de datos.

2) Grandes Bases de Datos y espacio de la dirección; Oracle apoyó la más grande Bases de Datos, potenciando el tamaño Terabytes, para hacer eficiente los recursos hardware expansivos, ello permite un control total del espacio usado.

3) Muchos usuarios concurridos en la Base de Datos; Oracle soporta un gran número de usuarios ejecutando una variedad de aplicaciones de Bases de Datos operando sobre los mismos datos. Ello minimiza la discusión de datos y garantiza la concurrencia de datos.

4) Alta transacción y prestación de procesamiento; Oracle mantiene las anteriores características con un alto grado de sistemas conjuntos de funcionamiento. Los usuarios de la Base de Datos no sufren desde el lento funcionamiento de procesamiento.

5) Alta disponibilidad; En algunos sitios Oracle trabaja 24 horas al día sin límite de tiempo para buscar a través de la Base de Datos.

6) Disponibilidad controlada; Oracle puede selectivamente controlar la disponibilidad de los datos, al nivel de la Base de Datos y al subnivel de la Base de Datos. Por ejemplo, un administrador puede rechazar el uso de una aplicación específica, así que los datos de la aplicación puedan ser recargados, sin afectar a otras aplicaciones.

7) Abrir industrias estándar; Oracle adhiere aceptar industrias estándar, para el lenguaje de acceso de datos, sistemas operativos, usuarios de interfaces, y protocolos de comunicación network. Esto es un sistema abierto que protege la inversión de los clientes.

8) Seguridad dirigida; para proteger contra el acceso y uso a Bases de Datos inautorizadas, Oracle proporciona un debilitamiento seguro para limitar las características y el acceso a datos de monitor. Estas características hacen fácil dirigir siempre el más complejo diseño para el acceso de datos.

9) La Base de Datos ha de cumplir la integridad; Oracle cumple con la integridad de los datos, “normas de negocios” que dictan estos estándares para aceptar los datos. Como un resultado, el coste de los códigos y dirigimiento comprobado en muchas aplicaciones de Bases de Datos serán eliminados.

10) Sistemas distribuidos; Para network, entornos distribuidos, Oracle combina los datos físicos localizados en diferentes computadoras dentro de la Base de Datos lógica que puede ser accedida por todos los usuarios network. Sistemas distribuidos tienen el mismo grado de usuarios, transparencia y consistencia de datos como sistemas no distribuidos, ya que recibe las ventajas de la dirección de la Base de Datos local. Oracle también ofrece la opción de heterogeneidad que permite a los usuarios acceder a los datos sobre algunas Bases de Datos transparentes que no son Oracle.

11) Portabilidad; El software Oracle es un puerto para trabajar bajo diferentes sistemas operativos. Aplicaciones desarrolladas para Oracle pueden ser portadas a algunos pequeños sistemas operativos o no modificación.

12) Compatibilidad; El software Oracle es compatible con industrias estándar, incluidas más industrias estándar de sistemas operativos. Aplicaciones desarrolladas para Oracle pueden usarse en prácticamente cualquier sistema con poco o sin modificación.

13) Conectividad; El software Oracle permite diferentes tipos de computadoras y sistemas operativos para dividir información a través de networks.

14) Réplica de entornos; El software Oracle te permite replicar grupos de tablas y su complemento de apoyo para múltiples sitios. Oracle soporta la replica de ambos datos y nivel de esquema. La tecnología de replicación flexible de Oracle soporta un sitio básico principal de replicación así como avanzado dinámico y posesión compartida de modelos.

El Servidor Oracle

El servidor Oracle es un objeto relacional del sistema de dirección que proporciona un abierto extenso, e integrado acceso para la dirección de la información. Un servidor Oracle consiste en una Base de Datos Oracle y un caso de servidor Oracle.

SQL (STRUCTURED QUERY LANGUAGE)

SQL (pronunciado SEQUEL) es el lenguaje de programación que define y manipula la Base de Datos. SQL es una Base de Datos relacional; esto significa simplemente que los datos son almacenados en una grupo de relaciones simple. Una base de datos puede tener una o más tablas. Y cada tabla tiene columnas y filas. Una tabla que tiene una base de datos Empleado, por ejemplo, debe tener una columna llamada número de empleado y cada fila en esa columna tendría un número determinado de empleados.

Puedes definir y manipular datos en una tabla con la orden SQL. Tu usas el lenguaje de definición de datos (DDL), orden para establecer el dato. La orden DDL incluye la orden para crear y cambiar las bases de datos y las tablas.
Puedes poner al día, suprimir o recuperar un dato en una tabla con el comando de manipulación de datos (DML). La orden DML incluye la orden para cambiar y buscar datos. El más común en SQL es la orden SELECT, el cual permite suprimir datos desde la base de datos.

En adición al comando SQL, el servidor Oracle tiene un lenguaje procedimental llamado PL/SQL. PL/SQL permiten al programador programar el estado en SQL. Esto permite controlar el curso de un programa SQL, para usar variables, y para escribir error de manejo.

8.1.2 Estructura de la Base de Datos y espacio de la dirección

Una base de datos Oracle tiene una estructura Física y una estructura lógica. Porque el servidor de la estructura física y lógica son separadas, el almacén físico de datos puede ser dirigido sin afectar el acceso al almacenamiento de la estructura lógica.

Estructura de una base de datos física

Una estructura de una base de datos física en Oracle está determinada por el sistema operativo de ficheros que constituye la base de datos. Cada base de datos Oracle está hecha de tres tipos de ficheros: uno o más ficheros de datos, dos o más ficheros para registrar el REDO, y uno o más ficheros de control. El fichero de una base de datos Oracle proporciona el actual almacenamiento físico para la información de bases de datos.

Estructura de una base de datos lógica

Una estructura lógica de bases de datos Oracle está determinada por:

- Una o mas tablas de espacio (tablespaces). Una tabla de espacio es un área lógica de almacenamiento.
- El objeto del esquema de la base de datos. Un esquema es una colección de objetos. Objeto esquema son las estructuras lógicas que directamente refieren a los datos de la base de datos, incluye también estructuras como tablas, vistas, secuencias, almacenamiento de procedimiento, sinónimos, índices, clusters, y lincado de bases de datos.

El almacenamiento de estructuras lógicas, incluyendo tablespaces, segmentos, y extensiones dictan como el espacio de una bases de datos es usada. El objeto esquema y las relaciones sobre sus formas del diseño relacional de una base de datos.

Base de datos Oracle

Una base de datos Oracle es una colección de datos que es tratado como una unidad. El propósito general de una base de datos es almacenar y recuperar información relacionada.
La base de datos tiene una estructura lógica y una estructura física.

Abrir y cerrar una base de datos

Una bases de datos Oracle puede ser abierta (accesible) o cerrada (no accesible). En situaciones normales, la base de datos es abierta y disponible para usar. Así, la base de datos es a veces cerrada por funciones especificas administrativas que requiere los datos de la base de datos para ser indisponible.

8.1.3 Estructura de la Memoria y Procesado

En esta sección discutiremos la estructura de la memoria y procesamiento usados por un servidor Oracle para dirigir una base de datos. Entre otras cosas, también discutiremos las características de la arquitectura, que proporciona un entendimiento de las capacidades del servidor Oracle para soportar:

- Muchos usuarios a mismo tiempo acceden a una única base de datos.
- El alto rendimiento requerido para concurrir sistemas multi-usuarios, multi-aplicación.

Un servidor Oracle usa estructuras de memoria y procesamiento para dirigir y acceder a la base de datos. Todas las estructuras de memoria existen en la memoria principal de las computadoras que constituyen el sistema de base de datos.
Procesamiento son los trabajos o tareas que trabajan en la memoria de estos computadores.

Estructura de la memoria

Oracle crea y usa estructuras de memoria para completar varios trabajos. Por ejemplo, la memoria es usada para almacenar códigos de programa siendo ejecutados y datos que son compartidos entre usuarios. Varias estructuras básicas de memoria están asociadas con Oracle; (SGA) el área global de sistema (el cual incluye el buffer de la base de datos), y el área global de programa (PGA).

El área global de sistema (SGA) es una región de memoria compartida que contiene datos e información de control para un caso Oracle. Un SGA y el proceso secundario Oracle constituyen un caso Oracle. Oracle reparte el área global de sistema cuando un caso comienza y se asigna cuando el caso se cierra. Cada caso tiene su propio área global de sistema.

Los usuarios normalmente conectan con un servidor Oracle dividiendo los datos en el área global de sistema. Para un óptimo rendimiento, el área global de sistema entero debería ser tan grande como sea posible, (mientras todavía esté instalado en la memoria) para almacenar tantos datos en memoria como sea posible y minimizar el disco I/O.

La información almacenada sin el área global de sistema está dividida en varios tipos de estructuras de memoria incluyendo el buffers de la base de datos, y el buffer Redo. Estas áreas tienen fijas los tamaños y son creadas durante el caso comienza.

El buffer Redo del área global del sistema almacena entradas Redo menos un logaritmo de cambios hechos a la base de datos. Las entradas Redo almacenadas en el buffers Redo están escritas en línea en ficheros Redo, lo cual es usado si la recuperación de la base de datos es necesaria. Su tamaño es estático.

El (PGA) área global de programa es un buffer de memoria que contiene datos e información de control para un proceso servidor. Un PGA es creado por Oracle cuando un proceso servidor es comenzado. La información en un PGA depende de la configuración de Oracle.

Ejemplo de cómo trabaja Oracle

El siguiente ejemplo ilustra una configuración Oracle donde el usuario y el proceso servidor asociado están en máquinas separadas (conectadas a una red).

1. Un caso es actualmente ejecutado en la computadora que está ejecutando Oracle (a menudo llamado Host o servidor de la base de datos).
2. Un computador ejecuta una aplicación (una máquina local o estación de trabajo cliente), en un proceso de usuario. La aplicación cliente intenta establecer una conexión al servidor usando el propio drivers Net8.
3. El servidor está ejecutando el propio drivers Net8. El servidor detecta la conexión requerida desde la aplicación y crea un proceso servidor en nombre del proceso usuario.
4. El usuario ejecuta una declaración SQL y entrega la transacción. Por ejemplo, el usuario cambia un nombre en una fila de la tabla.
5. El proceso servidor recibe la declaración y comprueba la parte junta para algún área SQL que contiene una declaración idéntica en SQL. Si un área SQL dividida es encontrada, el proceso servidor comprueba que el usuario tiene un acceso privilegiado para el dato requerido y el previamente existente área SQL dividida, es usado para declaración de procesos; sino, un nuevo área SQL dividida es asignado por la declaración así que si puede ser procesado.
6. El proceso servidor recupera algún valor de dato necesario desde el fichero de datos actual (tabla) o aquellos almacenados en el área global de sistema.
7. El proceso servidor modifica los datos en el área global de sistema. El DBWR (escribir una base de datos) es un proceso que escribe bloques modificados permanentemente al disco cuando hacerlo así es eficiente. Porque la transacción entregada, el proceso LGWR (escribir el registro) registra inmediatamente la transacción en el conectado de ficheros Redo.
8. Si la transacción es exitosa, el proceso servidor envía un mensaje a través de toda la red a la aplicación. Si no es exitosa, un mensaje de error es transmitido.
9. Por todo esto, todo procedimiento, la ejecución de otros procesos secundarios, mirando para condiciones que requieren intervención. En adición, el servidor de la base de datos dirige otras transacciones de usuario y previenen discusiones entre transacciones que requieren el mismo dato.

8.1.4 Concurrencia de los datos y consistencia de los datos

Aquí se explica el mecanismo software usado por Oracle para desempeñar los siguientes requisitos de un sistema de dirección de la información:

- Los datos deben ser leídos y modificados de una manera consistente.
- La concurrencia de los datos de un sistema multi-usuario deben ser maximizados.
- Alto rendimiento requerido para maximizar la productividad desde los muchos usuarios del sistema de la base de datos.

8.1.5 Procesamiento distribuido y Bases de datos distribuidas

Como un computador en red llega a ser más y más dominante en los entornos de la informática de hoy, los sistemas de dirección de bases de datos deben ser capaces de tomar ventajas de procesamiento distribuido y capacidad de almacenamiento. Esta sección explica las características arquitectónicas de Oracle que reúne estos requisitos.

Arquitectura cliente/servidor: procesos distribuidos

Los procesos distribuidos usan más de un procesamiento para dividir el proceso por un grupo de trabajos afines. Los procesos distribuidos reducen la carga de procesos en un único procesador para permitir diferentes procesos, para concentrar en un subgrupo de tareas afines, así proporcionan la realización y capacidad del sistema como un todo.
Un sistema de base de datos Oracle puede fácilmente tener ventajas de procesos distribuidos por usar su arquitectura. En esta arquitectura, el sistema de bases de datos está dividido en dos partes: una parte final de frente o porción cliente y una parte final de espalda o porción de servidor.

La porción cliente es la aplicación parte final de frente e interactuan con un usuario a través de la visualización, y señala los recursos tal como un ratón. La porción cliente no tiene responsabilidades al acceso de datos; ello se concentra en requerir, procesar y presentar datos dirigidos por la porción servidor. La estación de trabajo cliente puede ser optimizada por sus trabajos. Por ejemplo, debería no necesitar una capacidad de disco grande o debería de beneficiar las capacidades gráficas.

La porción servidor la ejecuta el software Oracle y maneja las funciones requeridas para la concurrencia, acceso a datos compartidos. La porción servidor recibe y procesa SQL y PL/SQL, estado originado desde la aplicación cliente. El computador dirige la porción de servidor que puede ser optimizada por sus derechos, por ejemplo puede tener una capacidad de disco grande y procesadores rápidos.

Bases de datos distribuidas

La información en esta sección con respecto a puestas al día distribuidas y dos fases cometidas aplicadas solamente a sistemas que usa Oracle con la opción distribuida.

Una base de datos distribuida es una red de bases de datos dirigida por múltiples servidores de bases de datos que aparecen a un usuario como una única base de datos lógica. Los datos de todas las bases de datos en la base de datos distribuida pueden ser simultáneamente accedidos y modificados. El principal beneficio de una base de datos distribuida es que los datos de la base de datos física separados de la base de datos pueden ser combinadas lógicamente y potencialmente, hechas accesibles a todos los usuarios en una red.

Cada computador dirige una base de datos, en la base de datos distribuida es llamada nodo. La base de datos a la cual un usuario está directamente conectado es llamado base de datos local. Algún acceso a la base de datos adicional por este usuario son llamadas base de datos remotas. Cuando una base de datos local accede a la base de datos remota por información, la base de datos local es un cliente del servidor remoto.

Mientras una base de datos distribuida permite aumentar el acceso a una gran cantidad de datos a través de una red, ello debe además proporcionar la habilidad para ocultar la localización de los datos y la complejidad de acceder a la red. El DBMS distribuido debe además preservar las ventajas de administrar cada base de datos local como si ello fuera no distribuido.

Oracle y Net8

Net8 es un mecanismo Oracle para interfaces con el uso de protocolos de comunicación por la red, que facilita procesos distribuidos y bases de datos distribuidas. Los protocolos de comunicación definen el camino que los datos están transmitiendo y se reciben en una red. En un medio ambiente de red, un servidor de bases de datos Oracle se comunica con la estación de trabajo cliente y otros servidores de bases de datos Oracle usan el software Oracle para llamar a Net8.

Net8 soporta las comunicaciones en todo el protocolo de la red, PC/LAN soporta un grupo de estos para ser usados por grandes sistemas de computadores mainframe.

Usando Net8, el desarrollo de la aplicación no tiene que preocuparse de soportar la comunicación de la red en una aplicación de bases de datos. Si se usa un nuevo protocolo, el administrador de la base de datos hace algunos cambios menores, mientras la aplicación no requiera la modificación y continúe con su función.

8.1.6 Comenzar y acabar las operaciones

Una base de datos Oracle no está disponible a los usuarios hasta que el servidor Oracle ha comenzado y la base de datos haya sido abierta. Estas operaciones deben cumplirse por el administrador de la base de datos. Comenzar una base de datos y hacer esto disponible por el amplio sistema utiliza estos tres pasos:

1. Comenzar un caso del servidor de la base de datos.
2. Montar la base de datos.
3. Abrir la base de datos.

Cuando el servidor Oracle ha comenzado, este usa un fichero parámetro que contiene los parámetros de inicialización. Estos parámetros especifican el nombre de la base de datos, la cantidad de memoria para reparar, el nombre de los ficheros de control, y varios límites y otros parámetros del sistema.

Cerrar un caso y la base de datos que es conectada toma tres pasos:

1. Cerrar la base de datos.
2. Disminuir la base de datos.
3. Cerrar el caso del servidor Oracle.

8.1.7 Seguridad de la Base de Datos

Un sistema de base de datos multi-usuario, semejante como Oracle, incluye características de seguridad de como una base de datos es controlada, usada y accedida. Por ejemplo, los mecanismos de seguridad que utiliza son los siguientes:

1. Prevenir el acceso inautorizado de la base de datos.
2. Prevenir el acceso inautorizado a objetos esquema.
3. Controlar el uso de discos.
4. Controlar el uso de los recursos del sistema (tiempo de CPU).
5. Auditorias de acciones de usuarios.

Asociado con cada usuario de la base de datos está el esquema por el mismo nombre. Un esquema es una colección lógica de objetos (tablas, vistas, secuencias, sinónimos, índices, clusters, procedimientos, funciones, paquetes y lincado de bases de datos). Por el contrario cada usuario crea y tiene acceso a todos los objetos en el correspondiente esquema.

La seguridad de la base de datos puede ser clasificada dentro de dos categorías: sistemas de seguridad y seguridad de datos.

Sistema de seguridad incluye el mecanismo que controla el acceso y uso de la base de datos al nivel del sistema. Por ejemplo, el sistema de seguridad incluye:

- Validar la combinación del nombre del usuario/password.
- La cantidad de espacio en disco disponible para los objetos de un usuario.
- El límite de recurso para un usuario.

El sistema de seguridad controla los mecanismos:

- Si un usuario es autorizado para conectarse a la base de datos.
- Si la auditoria de la base de datos es activa.
- Si las operaciones del sistema son realizadas por un usuario.

Seguridad de datos incluye el mecanismo que controla el acceso y uso de la base de datos al nivel del objeto.

Por ejemplo, seguridad de datos incluye:

- Que el usuario tiene acceso a un esquema objeto específico y el tipo específico de acción que permite cada usuario en el objeto (por ejemplo, usuario SCOTT puede cuestionar las declaraciones SELECT y INSERT pero no el DELETE usando la tabla EMP).
- Las acciones, si tiene un auditor por cada objeto esquema.

Mecanismos de seguridad

El servidor Oracle proporciona un acceso de control discreto, que es menor el acceso restringido a la información basada en privilegios. El privilegio apropiado debe ser asignado a un usuario, en el sentido que el usuario tenga acceso a un objeto. El privilegio apropiado para un usuario puede asentir al privilegio de otros usuarios a su discreción; por esta razón, este tipo de seguridad es llamada “discreta”.

La seguridad de la dirección de la base de datos Oracle usando diferentes tipos:

- Usuarios de la base de datos y esquemas.
- Privilegios.
- Roles.
- Colocación y fijación del almacenamiento.
- Limite de recursos.
- Auditando.

Dominios de seguridad

Cada usuario tiene un dominio de seguridad, un juego de propósitos que determina semejante cosa como la:

- Acción (privilegios y roles) disponibles para el usuario.
- Cuotas del espacio de las tablas (espacio disponible en disco) para el usuario.
- Límites de los recursos del sistema (por ejemplo, tiempo de procesamiento de la CPU) para el usuario.

Privilegios

Un privilegio es un derecho a ejecutar un tipo particular de declaraciones SQL. Como por ejemplo incluir privilegios de:

- Derecho a conectar a la base de datos (crear una sesión).
- Derecho a crear una tabla en tu esquema.
- Derecho a seleccionar las fichas de algunas tablas.
- Derecho a ejecutar algunos procedimientos almacenados.

El privilegio de una base de datos Oracle puede ser dividido en dos distintas categorías; privilegios de sistema y privilegio de objetos.

Privilegios de sistema

El privilegio de sistema permite al usuario realizar una acción particular en un gran sistema o acción particular en un tipo particular de objetos. Por ejemplo, el privilegio creado en una tabla o el borrado de las filas y una tabla en la base de datos es privilegio de sistema. Muchos privilegios de sistemas están disponibles solo a administradores y desarrollo de aplicaciones porque el privilegio es muy poderoso.

Privilegios de objetos

El privilegio de objetos permite a los usuarios realizar una acción particular en un objeto esquema específico. Por ejemplo, el privilegio a borrar filas de una tabla especifica es un privilegio objeto. El privilegio objeto es asignado a usuarios finales así que ellos pueden usar una aplicación de la base de datos a terminar una tarea específica.

8.1.8 Bases de Datos de reserva y recuperación

Esta sección cubre las estructuras y mecanismos de software usados por Oracle para proporcionar:

- La base de datos requerida para recuperar los diferentes tipos de fallo.
- Recuperar operaciones flexibles para una situación conjunta.

- Disponiendo de datos durante la recuperación de operaciones así que el usuario del sistema puede continuar su trabajo.

El modelo objeto relacional

El modelo objeto relacional permite al usuario definir tipos de objetos, especificando ambos la estructura de los datos, el método de operación en los datos, y el uso de los tipos de datos con el modelo relacional.

Los tipos de objeto son abstracciones del mundo real, de entidades ejemplo, programas de aplicaciones de ordenes con pacto. Un tipo de objeto tiene tres tipos de componentes:

- Un nombre, que sirve para identificar el tipo de objeto único.
- Atributos, que son empotrados en el tipo de datos o otros tipos definidos por el usuario. La estructura de modelos de atributos de las entidades del mundo real.
- Métodos, que son funciones o procedimientos escritos en PL/SQL y almacenados en la base de datos, o escritos en lenguaje C y almacenados externamente. Las operaciones de implementación de métodos específicos que una aplicación puede realizar en los datos. Cada tipo objeto tiene un método constructor que hace un nuevo objeto de acuerdo a la especificación de los tipos de datos.

Tablas

Una tabla es una unidad básica de almacenamiento de datos en una base de datos Oracle. Las tablas de una base de datos tienen todos los datos accesibles al usuario.

Los datos de las tablas son almacenados en filas y columnas. Cada tabla es definida con un nombre de tabla y juego de columnas. Cada columna tiene un nombre de columna, un tipo de dato (semejante a CHAR, DATE, o NUMBER), y un ancho (con el cual debe ser predeterminado por el tipo de dato, como es DATE) o escala y precisión (por el NUMBER solo tipo de dato). Una vez una tabla es creada, y válidas las filas de datos pueden insertarse. Las filas de tablas pueden ser preguntas, borradas o modificadas.

Vistas

Una vista es una costumbre de presentación de los datos en una o más tablas. Una vistas también puede ser opinión de cómo se almacenan las preguntas.

Las vistas no contienen actualizaciones o datos almacenados; antes, ellos derivan sus datos desde las tablas en que ellos están basados, referentes a las tablas base de la vista. Las tablas base pueden por turnos ser ellas mismas vistas.

Las vistas pueden ser usadas para lo siguiente:

- Proporcionar un nivel adicional de tablas de seguridad por accesos restringidos a un juego predeterminado de filas y columnas de una tabla. Por ejemplo, una vista de una tabla puede ser creada así que las columnas son datos sensitivos (por ejemplo, información del salario) no incluidos en la definición de las vistas.
- La complejidad de los datos ocultos. Por ejemplo, una única vista puede combinar 12 mensualidades de tablas de liquidación para proporcionar un año de datos para analizar e informar. Una única vista puede también ser usada para crear un juego, que es una visualización de columnas o filas en múltiples tablas. De cualquier manera, las vistas ocultan el hecho que estos datos originen actualizaciones de varias tablas.

- Simplificación de los comandos por el usuario. Por ejemplo, las vistas permiten al usuario seleccionar la información de múltiples tablas sin requerir al usuario saber actualmente como realizar una correlación de subpreguntas.
- Presentar los datos en una perspectiva diferente desde esa tabla. Por ejemplo, las vistas proporcionan unos significados para renombrar las columnas sin afectar a las tablas en las cuales está basada la vista.
- Almacenar preguntas complejas. Por ejemplo, una pregunta puede realizar cálculos extensos que informan sobre la tabla. Para salvar estas preguntas como una vista, los cálculos son realizados solo cuando la vista es preguntada.

Las vistas que implican un juego (una declaración SELECT que selecciona datos desde múltiples tablas) de dos o más tablas pueden solo actualizar ciertas condiciones.

8.1.9 El modelo relacional para la dirección de la Base de Datos

El sistema de dirección de la base de datos tiene que desarrollar una jerarquía para la red del modelo relacional. El más ancho modelo de bases de datos aceptadas es el modelo relacional. Oracle prolonga el modelo relacional para un modelo relacional objeto, que hace posible el modelo de negocios de almacenamiento complejo en una relación de base de datos.

8.1.10 Acceso a datos

Esta sección introduce como Oracle reúne los requisitos generales para un DBMS para hacer lo siguiente:

- Adherir a la industria estándar para un lenguaje de acceso a datos.
- Controlar y preservar la consistencia de una información de la base de datos mientras son manipulados sus datos.
- Proporcionar un sistema para definir y hacer cumplir normas para la integridad de la información de una base de datos.
- Proporcionar un alto funcionamiento.

SQL (Structured Query Language)

SQL es un simple, pero poderoso lenguaje de acceso a la base de datos que es el lenguaje estándar para dirigir el sistema de la base de datos relacional. El SQL implementado por la corporación Oracle para Oracle es un 100% de conformidad con el ANSI/ISO lenguaje de datos estándar SQL.

Estado de SQL

Toda la información de las operaciones en una base de datos es realizada usando el estado SQL. Un estado SQL es una cadena de texto SQL que está dando a Oracle para ejecutar. Un estado debe ser el equivalente de una sentencia SQL completa, como en:

SELECT ename, deptno FROM emp;

Solo un estado SQL completo puede ser ejecutado, mientras que un fragmento de sentencia, tal como las siguientes indican un error que requiere más texto ante un estado SQL puede ejecutar:
SELECT ename
Un estado SQL puede ser visto como un muy simple, pero poderoso programa de computador o instrucción.

Un estado SQL está dividido en las siguientes categorías:

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- Control del estado de la transacción.
- Estado del estado de la sesión.
- Estado del sistema de control.
- Estado de empotrar SQL.

PL/SQL

PL/SQL es un lenguaje de extensión procedimental Oracle. PL/SQL combina la facilidad y la flexibilidad de SQL con el funcionamiento procedimental de un lenguaje programado estructurado, tal como IF…THEN, WHILE, y LOOP.

Cuando designamos una aplicación de la base de datos, un promotor debería considerar las ventajas de usar el almacenamiento PL/SQL:

- Porque el código PL/SQL puede ser almacenado centralmente en una base de datos, tráfico de red entre aplicaciones y la reducción de la base de datos, así la aplicación y el funcionamiento del sistema aumenta.
- El acceso de los datos puede ser controlado por el código de almacenamiento PL/SQL. En estos casos, el usuario de PL/SQL puede acceder a los datos solo como deseado por la aplicación del promotor (al menos otra ruta de acceso es concedida).
- Los bloques PL/SQL pueden ser enviados por una aplicación a una base de datos, ejecutando complejas operaciones sin excesivo tráfico de red.

Si cuando PL/SQL no está almacenado en la base de datos, las aplicaciones pueden enviar bloques de PL/SQL de la base de datos al estado individual SQL, así otra vez se reduce el tráfico de la red.

BIBLIOGRAFÍA:

LIBRO:

DISEÑO DE BASES DE DATOS RELACIONALES;

Autores: Adoración de Miguel Castaño
Mario Gerardo Piattini Velthuis

Capítulo 24: Diseño Físico de Bases de Datos.

LIBRO:

CONNOLLY, II EDICIÓN.

Capítulo 9: Diseño Físico de Bases de Datos.
Apartado 4.3: Introducción al diseño de Bases de Datos.

REFERENCIAS WEB:

http://www.lsl.us.es/docencia/asignaturas/dbd.html
Departamento de Lenguajes y Sistemas Informáticos
Diseño de Bases de Datos
Tema 6, Diseño Físico de Bases de Datos

http://selene.uc3m.es/labda/Dabd.htm
Asignatura, Diseño Avanzado de Bases de Datos
Tema 5, Diseño avanzado en el modelo relacional.
Apartado 5.4, Diseño Físico.