CI-5313: Arquitectura y Administración de Base de Datos I
Apuntes del curso

Soraya Abad Mota

Versión 1: Septiembre 2002
Actualizaciones: Enero 2005 y Septiembre 2007

1. Objetivos del curso

Entender y ejercitar los aspectos físicos de un Sistema de Base de Datos, desde el diseño físico (modelo interno) hasta el día a día de las operaciones sobre la base de datos (transacciones), pasando por todas las labores del Manejador de Base de Datos (servicios).

Una abreviación que utilizaremos frecuentemente para el software Manejador de Base de Datos es DBMS, por sus siglas en inglés, *data base management system*.

2. Tópicos del curso

1. **Diseño Físico.** El eje central de la fase de Diseño Físico de una base de datos es tomar decisiones en cuanto a las organizaciones de archivo que se van a utilizar para almacenar los datos. Se trata en esta fase de seleccionar, de las organizaciones ofrecidas por el DBMS, las más adecuadas para el uso que se le va a dar a cada conjunto de datos, de modo que se "minimicen" los tiempos de acceso a los datos.

   El diseño físico es muy relevante cuando hay un gran volumen de datos a almacenar. Es una tarea que depende grandemente de lo que ofrezca el DBMS a utilizar. Lightstone, Teorey y Nadeau en su libro: "Physical Database Design" (Morgan Kaufmann 2007) dan lineamientos muy útiles para cuatro aspectos bien concretos en los cuales se pueden concentrar las decisiones de diseño físico para lograr un buen rendimiento de las transacciones que operan sobre la base de datos. Estos aspectos son: el uso de índices, el uso de vistas materializadas, el particionamiento y el "clustering" multidimensional y la desnormalización. La motivación fundamental para invertir esfuerzos en esta etapa son las afirmaciones de algunos diseñadores que han logrado mejorar la ejecución de algunas consultas críticas para que su tiempo de ejecución sea 50 veces más rápido que la ejecución "ingenua", sin diseño físico. En el laboratorio se ejercitan estos conceptos con un manejador de base de datos específico.

   Antes de poder entender las opciones en esta fase es necesario conocer el medio donde se almacenan los datos y las organizaciones de archivo y los métodos de acceso

a las mismas, temas que se cubren en el tópico de estructuras de almacenamiento secundario.

2. **Estructuras de almacenamiento secundario**. Para poder tomar decisiones sobre las estructuras físicas donde almacenar la base de datos es fundamental conocer sobre los medios de almacenamiento y las estructuras lógicas de almacenamiento secundario disponibles. El tratamiento de este tópico se divide en dos aspectos fundamentales: generalidades sobre los medios de almacenamiento y cómo se almacenan los datos en un medio, en particular en los discos magnéticos, y el estudio de las diferentes organizaciones de archivos en disco para almacenar los datos, de acuerdo a las operaciones que se deban hacer sobre los mismos. El tópico incluye: Medios, estrategias, niveles de abstracción y estructuras de almacenamiento, estos temas se puede encontrar en los dos libros básicos de texto, a saber:

   - Navathe y Elmasri (2004, cuarta edición), capítulos 13 (Disk Storage, Basic File Structures, and Hashing) y 14 (Indexing Structures for Files). Ahora hay una quinta edición de este mismo libro.
   - Ramakhrishnan y Gehrke (2003, tercera edición), Parte III: Storage and Indexing, en los siguientes capítulos: 8. Overview of Storage and Indexing (pp.273-303), 9.Storing Data: Disks and Files (pp.304-337), 10. Tree-structured indexing (pp. 338-369), y 11. Hash-based indexing (pp. 370-390). (NOTA: En la edición anterior del Ramakhrishnan y Gehrke (2000, segunda edición), este tópico se trata en el capítulo 7, páginas 195-229 y capítulo 8, páginas 230-246.)

   En este curso cubriremos este tópico usando como guía el Navathe y Elmasri, y complementando algunos aspectos con el Ramakrishnan y Gehrke. Como complemento adicional se puede consultar el libro de Jim Gray y Andreas Reuter, "Transaction Processing: Concepts and Techniques" de la bibliografía del curso, capítulo 13: File and Buffer Management.

3. **Optimización de Consultas.** En un lenguaje de alto nivel como SQL, el programador no puede indicar cómo se debe ejecutar una consulta. El procesador de consultas del DBMS tiene un mecanismo para representar la consulta y para encontrar un método de ejecución "óptimo". Este mecanismo utiliza la información contenida en el Diccionario de Datos acerca de cómo están almacenados los datos (por ejemplo, las relaciones en el modelo relacional) que usa la consulta e información acerca del tamaño de las estructuras donde están almacenados los datos, y otros parámetros, para decidir los mejores algoritmos de ejecución de la consulta. El libro de Navathe et al. trata este tópico en el capítulo 15 (Algorithms for Query Processing and Optimization). El libro de Ramakrishnan y Gehrke trata este tópico en la parte IV de Query Evaluation, capítulos: 12 Overview of query evaluation (pp.393-420), 14 Evaluating Relational Operators (pp.439-477) y 15 A Typical Relational Optimizer.

4. **Transacciones.** El concepto de transacción en base de datos y sus implicaciones sobre la concurrencia y la integridad de los datos es un tópico esencial.

5. **Control de Concurrencia.** El servicio de control de concurrencia que debe ofrecer un DBMS se ocupa de que la ejecución concurrente de varias transacciones no altere la integridad de los datos. Se utiliza la teoría de la "seriabilidad" (serializability) de las transacciones como base del control de concurrencia.

6. **Recuperación.** Ante la presencia de errores del sistema o de los medios de almacenamiento, el módulo de recuperación del DBMS se encarga de garantizar la integridad de los datos, en otras palabras, su objetivo es mantener siempre la base de datos en un estado consistente aún en la presencia de fallas.

7. **Seguridad e Integridad.** Cada usuario definido en la base de datos, está autorizado para acceder y operar sobre una porción de la misma. En este tópico se estudian los mecanismos que proveen los manejadores de base de datos para especificar los aspectos de seguridad del sistema de base de datos.

Los tópicos de transacciones, control de concurrencia y recuperación se cubren en el libro de Phil Bernstein, V. Hadzilacos y Nathan Goodman, "Concurrency Control and Recovery in Database Systems", el cual se encuentra disponible electrónicamente en la página de Phil Bernstein dispuesta para tal fin, a saber: http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx

Hay una interrelación muy fuerte entre este curso y el área de Sistemas de Operación. Vamos a estar usando constantemente conceptos de esa área y supondremos que el estudiante domina esos conceptos.

3. Tópico 1. Introducción y Conceptos Básicos.

En la figura 1 se muestra la Arquitectura de un Sistema de Base de Datos. En este curso nos concentramos en el nivel inferior de esta arquitectura, el más primitivo, el modelo interno. Nuestro gran aliado es el DBMS. Ahora sí pensamos en él desde el principio y en los servicios que nos debe proveer. Cada módulo o servicio del DBMS corresponde con un tópico de este curso.

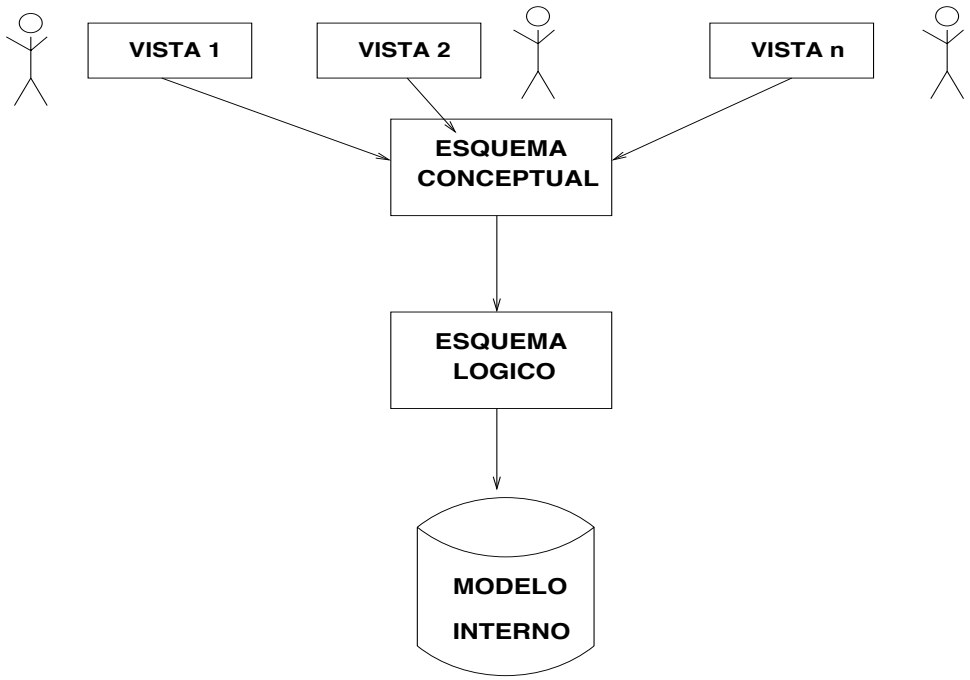

Figura 1: Arquitectura de un Sistema de Base de Datos

3.1. ¿Dónde se almacena la base de datos?

El almacenamiento en un computador es como una pirámide. En el tope de la pirámide están las memorias más rápidas, de menor capacidad y más costosas, como lo son la memoria principal y la memoria cache. En el medio de la pirámide se encuentran el almacenamiento secundario, o almacenamiento "en línea" (online), constituido principalmente por discos magnéticos y más recientemente incluye discos ópticos. El almacenamiento secundario es más lento, pero tiene mayor capacidad que el primario. Finalmente, en la base de la pirámide está el almacenamiento terciario o "fuera de linea" (offline), el cual tiene una gran capacidad, pero requiere de una intervención mecánica para transferirlo al secundario, por ejemplo. Generalmente se habla de cintas magnéticas en esta parte de la jerarquía.

Hay ciertas características del almacenamiento en un computador que conducen a almacenar la base de datos en almacenamiento externo o secundario, entre ellas podemos encontrar las siguientes:

- Capacidad: la memoria principal tiene una capacidad limitada, su tamaño es menor que el necesario para almacenar una base de datos, en varios órdenes de magnitud.

- Economía: aún cuando se lograran capacidades de almacenamiento en almacenamiento primario del mismo orden del externo, el costo por byte sería mucho más alto.
- Durabilidad: la memoria principal (o primaria) es volátil, la externa (o secundaria) es "permanente". La principal se puede hacer durable con baterías, pero esto le aumenta el costo por byte.
- Velocidad: la velocidad del almacenamiento secundario es varios órdenes de magnitud menor que la velocidad del almacenamiento primario, y está mas lejos (latency and path length).
- Funcionalidad: los datos no pueden ser procesados en almacenamiento secundario, se deben llevar a memoria principal para poder operar sobre ellos. Por otro lado, para poder tener datos que perduren, los datos no se pueden dejar en memoria principal, en consecuencia, hay que ponerlos en almacenamiento externo.

La capacidad, la economía y la durabilidad son los aspectos de la memoria principal que hacen elegir los discos como medio de almacenamiento de la base de datos. La velocidad y la funcionalidad son las desventajas de usar los discos para este fin. El DBMS entonces debe buscar mecanismos para paliar estas desventajas, por ejemplo, tratando de que los datos que necesita una operación estén en memoria principal, antes de que esa operación se vaya a realizar.

3.2. Medios de almacenamiento y sus características

El principal medio de almacenamiento secundario que se sigue utilizando para almacenar las bases de datos es el disco magnético. Un disco está compuesto físicamente por *superficies (surface)* magnetizables las cuales están divididas en *pistas (track)*, las cuales a su vez se componen de *sectores (sector)*, todos los anteriores son conceptos físicos. Adicionalemente, existe la noción de *cilindro (cylinder)*, constituidos por la misma pista en todas las superficies del disco. Además, cada superficie tiene una cabeza lectora escritora que se mueve en las pistas y las superficies dan vueltas para que la cabeza tenga acceso a los distintos sectores de una pista.

Por otro lado está el concepto lógico asociado a un disco, a saber: *bloque de disco (block) o página (page)* el cual se define como la unidad básica en la cual se divide un sector de un disco, en los bloques se almacenan los datos que se guardan en los discos. Al formatear o incializar un disco se establecen los bloques del mismo, todos los bloques son de un mismo tamaño y también se deja un espacio de longitud fija entre bloques, el cual se llama *interblock gap*.

Debe existir una fuerte interacción entre el sistema de base de datos y el sistema básico de archivos (basic file system) del sistema de operación sobre el cual reside la base de datos. Generalmente se establece una jerarquía de correspondencias entre estructuras del Sistema de Base de Datos y estructuras del sistema básico de archivos. Las estructuras provistas por ambos sistemas constituyen los niveles de abstracción que permiten a los usuarios de la base de datos hacer referencia a estructuras lógicas (por ejemplo, una relación en el modelo relacional) y que el DBMS y el S.O. haga la correspondencia entre esa estructura lógica y los bloques de disco donde los datos de esa estructura están almacenados.

3.3. Registros y archivos

La unidad lógica en la cual se almacenan los datos de una base de datos es un archivo. Un archivo es un conjunto de bloques. La unidad fundamental dentro de un archivo es un registro, un registro contiene campos cada uno de los cuales tiene una longitud y admite un valor de un tipo determinado. La longitud del registro es la suma de las longitudes de sus campos. Los registros pueden ser de dos tipos: de longitud fija, cuando todos los registros tienen la misma longitud, o de longitud variable, cuando diferentes registros pueden tener diferentes longitudes. ¿Por qué pueden existir registros de longitud variable? por varias razones:

- algunos campos pueden tener longitud variable, por ejemplo, el nombre de una persona;
- por la existencia de *repeating fields* en el registro, es decir, campos que admiten varios valores, esto es análogo al caso de los atributos multivaluados en el modelo ER;
- algunos campos pueden ser opcionales, es decir, pueden tener el valor nulo y no ocupar espacio en el registro, o
- en el archivo hay registros de distinto tipo, por ejemplo como en el caso de la organización mixta que veremos más adelante.

Como la unidad de transferencia entre el disco y la memoria es el *bloque de disco*, los registros de un archivo se colocan en bloques de disco y es importante saber cuántos registros caben en cada bloque, a esto se le llama el *factor de bloqueo del archivo (blocking factor o bfr)* y se utiliza para luego calcular cuántos bloques de disco se necesitan para almacenar el archivo completo.

Para registros de longitud fija, si llamamos T al tamaño del bloque y l a la longitud del registro del archivo, y $T \geq l$ entonces, $bfr = \lfloor T/l \rfloor$ registros por bloque caben en este disco. Una vez conocido el bfr, si el número total de registros del archivo es r, para calcular el número de bloques necesarios para almacenar el archivo se usa: $\lceil r/bfr \rceil$. Para archivos con registros del longitud variable, el bfr sólo da el promedio de registros que caben en un bloque si en l se expresa la longitud promedio del registro. También se puede calcular el factor de bloqueo máximo, si en l se registra el valor de la longitud máxima del registro.

Como al calcular el factor de bloqueo la división puede no ser exacta, puede sobrar espacio en un bloque donde no cabe un registro completo; en este caso tenemos dos opciones para colocar los registros en los bloques de disco, estas opciones son: "spanned" u "unspanned". En el caso "spanned" se van almacenando registros del archivo en el bloque y cuando se acabe el espacio libre del bloque se continua en otro bloque, si un registro se comenzó a almacenar en un bloque y no cupo completo, se deja esa porción del registro allí y el resto del registro se almacena en el siguiente bloque disponible. De esta forma algunos registros pueden quedar almacenados en varios bloques, lo cual hace un poco mas complejo el almacenamiento de los registros en los bloques, pero ahorra espacio, pues no hay desperdicio. En el caso "unspanned", se colocan tantos registros en el bloque como quepan completos, si sobra espacio y no cabe un registro completo en ese espacio, no se utiliza y se sigue almacenando en el siguiente bloque.

Una vez definido cómo almacenar los registros de un archivo en los bloques de disco, queda por definir cómo relacionar todos los bloques de disco que pertenecen al mismo archivo. Hay varias alternativas para ello, a saber:

Ubicación contigua: todos los bloques del archivo se colocan en bloques consecutivos de disco. Esto hace más fácil recorrer todo el archivo, pero hace más difícil expandirlo.

Ubicación enlazada: cada bloque tiene un apuntador al siguiente bloque del archivo. Esto facilita el crecimiento del archivo, pero hace más difícil el recorrerlo completo.

"Clusters" de bloques consecutivos: se agrupan los bloques del archivo en "clusters", también llamados segmentos o "extents" y se enlazan los clusters con apuntadores. Este es el caso más general de la ubicación enlazada.

Ubicación indizada: se crean uno o más bloques de índices que tienen los apuntadores a los verdaderos bloques del archivo. Facilita el acceso directo a cada bloque del archivo, pero dificulta el recorrido completo del archivo.

Todo archivo tiene un encabezado o "header" que contiene la descripción del archivo, necesaria para los programas que acceden a los registros del archivo. La **información que se encuentra en el "header"** se puede resumir de la siguiente forma:

- información para determinar las direcciones en disco de los bloques del archivo;
- descripción del formato de los registros, por ejemplo, para registros de longitud fija y unspanned, se pueden almacenar la longitud de cada campo y el orden de los campos, y para registros de longitud variable, se pueden almacenar: los códigos de los tipos de campos, los caracteres de separación entre campos y los códigos de los tipos de registro;
- tamaño del archivo, número de registros, número de bloques, etc.

¿Cómo se busca un registro en el disco?

- Si se sabe la dirección del registro, se copia el bloque indicado por la dirección física, a un *buffer* de memoria, luego, usando la información del encabezado del archivo se ubica exactamente el registro en ese bloque. Si la colocación de los registros del archivo en los bloques es "spanned", el registro puede estar ubicado en más de un bloque y sería necesario traerse todos esos bloques a memoria para poder tener el registro completo.
- Si no se sabe la dirección del bloque donde está el registro, hay que buscar en todos los bloques y para cada bloque se copia a memoria, se recorre a ver si el registro está allí, hasta que se encuentre el registro o hasta que se hayan revisado todos los bloques y no se encuentre el registro.