Capítulo 1. Introducción.

Base de datos: Conjunto de datos que pertenecen a un mismo contexto y son almacenados sistemáticamente.

SGBD: Conjunto de datos interrelacionados, conjunto de programas para acceder a dichos datos.

- ***Objetivo***: almacenar y recuperar información de manera rápida y eficiente.

BD vs Sistemas de archivos:

- Diferentes programas y archivos para almacenar información (Redundancia e inconsistencia, difícil acceso, aislamiento, integridad, atomicidad, anomalías en el acceso concurrente, seguridad).

Abstracción de datos:

- ***Nivel físico:*** como se almacenan los datos
- ***Nivel lógico:*** que datos se almacenan y como se relacionan.
- ***Nivel de vistas:*** parte de los datos, dependiendo necesidades.

Modelos de datos:

- ***Modelo E-R:*** percepción del mundo real, mediante entidades y relaciones.
- ***Modelo Relacional:*** conjunto de tablas (F y C)

Lenguajes de BD:

- ***Definición de datos:*** esquema de BD
- ***Manipulación de datos:*** recuperación, inserción, borrado y modificación de la información de la BD

Capítulo 2. Procedimientos almacenados, funciones y triggers.

PA: grupo de sentencias SQL que forman una unidad lógica y ejecutan una tarea en particular. Encapsulan un conjunto de operaciones o consultas a ser ejecutadas. Pueden tener parámetros de I/O y de I O. Diferencia con funciones: funciones devuelven un valor único y los PA no. Características:

- Cumplir tarea específica
- No duplicar la funcionalidad de Oracle

Ventajas:

- ***Seguridad:*** permiso de ejecución a PA no a tablas
- ***Rendimiento:*** reduce tráfico de red, no necesita compilación en tiempo de ejecución.
- ***Asignación de memoria:*** basta cargar una copia en memoria.
- ***Productividad:*** evita duplicar código
- ***Integridad:*** no se necesita recompilar apps, testing, etc.

Creación:

- Compila el PA
- Almacena el código compilado en memoria
- Almacena el PA en la BD

```
CREATE [ OR REPLACE ] PROCEDURE [ schema. ]procedure
   [ (argument [ { IN | OUT | IN OUT } ]
               [ NOCOPY ]
               datatype [ DEFAULT expr ]
      [, argument [ { IN | OUT | IN OUT } ]
                  [ NOCOPY ]
                  datatype [ DEFAULT expr ]
      ]...
     )
   ]
   { IS | AS }
   BEGIN

      SQL SENTENCES

END;
```

Triggers: objeto con nombre de la BD asociado a una tabla, se activa/ejecuta cuando ocurre un evento sobre dicha tabla. Por:

- Sentencias DML/ DDL sobre la tabla ejecutadas por usuario
- Eventos de la BD: login, errores, arranque o parada del servidor, etc.

Pueden ser escritos en SQL, PL/SQL y java. Son similares a los PA.

- Procedimiento: app, usuarios, trigger
- Trigger: evento independiente del usuario o app

Capítulo 3. Indexación y asociación

Conceptos básicos: mecanismos para acelerar el acceso a los datos

- ***Clave de búsqueda***: atributo, para buscar registros en un archivo
- ***Archivo de índices***: consta de registros (entradas de índice) de la forma: CB | Puntero, más pequeños que archivo original.
- ***Clases de índices***: ordenados (CB ordenadas), asociativos (CB distribuidas en "cajones" con una h).

Métricas de evaluación: tipos de acceso soportados eficientemente. Ex. 1. Registros con un valor concreto en el atributo o 2. 1 con un rango de valores. Métricas: Tiempo de acceso, inserción, borrado costes de espacio.

Índices ordenados: evaluadas en base a:

- ***Índice ordenado***: entradas de índices ordenadas sobre una CB.
- ***Índice primario***: en un archivo ordenado secuencialmente, índice cuya CB determina orden del archivo. Se denomina, índice con agrupación (clustering). La CB es generalmente pero no necesariamente la clave primaria.
- ***Índice secundario:*** índice cuya CB determina un orden diferente al secuencial (sin agrupación).
- ***Archivo secuencial indexado:*** archivo secuencial ordenado con un índice primario.

Archivos de índice denso: registro del índice que aparece por cada valor de CB del archivo.

Archivos de índice disperso: registros del índice para algunos valores de CB. Cuando registros están ordenados secuencialmente.

- Localizar registro:
  - Encontrar el registro de índice con mayor valor de CB.
  - Búsqueda secuencial en el archivo.
- Menos espacio y menos costes de mantenimiento para inserciones y borrados.
- Más lento para localizar registros que el denso.
- Buen equilibrio: índice disperso con una entrada de índice por cada bloque en el archivo, correspondiente al menor valor de CB en el bloque.

Índice multinivel:

- Si el índice primario no encaja en memoria, acceso es costoso.
- Reducir accesos a disco. Mantener índice primario sobre disco como archivo secuencial y hacer índice disperso sobre él.
  - I. Externo: índice disperso del índice primario
  - I. Interno: archivo de índice primario.
- Si el índice externo es demasiado grande, se puede crear otro nivel de índice.
- Al insertar o borrar del archivo, se deben actualizar los índices a todos los niveles.

Actualización del índice: Borrado

- Si el registro borrado era el único con su valor de CB, la CB se borra.
- Borrado del índice de un solo nivel:
  - Densos: el borrado de CB es similar al borrado del registro.
  - Dispersos: si hay una entrada para CB en el índice que se borra, reemplazando la entrada en el índice con el siguiente valor CB. Si el valor de la siguiente CB tiene una entrada, se borra la entrada en vez de reemplazarla.

Actualización del índice: Inserción

- ***Inserción de índices de un solo nivel:***
  - Búsqueda con el valor de CB en el registro a insertar.
  - Densos: si el valor de CB no aparece en índice, insertarlo
  - Dispersos: si el índice almacena una entrada por cada bloque de archivo, no hacer cambios al índice a menos que se cree un nuevo bloque. Se inserta en el índice el primer valor de CB en el nuevo bloque.
- Algoritmos de inserción multinivel son extensiones de los de un nivel.

Índices secundarios:

- Encontrar registros cuyos valores en un campo cumplen una condición.
- Tener índice secundario con un registro del índice por cada valor de CB; el registro del índice apunta a un cajón con punteros a los registros actuales, con ese valor de CB.

Índices primario y secundario:

- Secundarios densos.
- Los índices ofrecen ventajas en la búsqueda
- Si se modifica archivo, actualizar índices (implica sobrecargas en la modificación de la BD)
- La búsqueda secuencial con índices primarios es eficiente, con un índice secundario es costosa (cada acceso al registro puede coger un nuevo bloque del disco)

Archivos de índice del árbol B+: alternativa a archivos secuenciales indexados.

- ***Inconvenientes de archivos secuenciales indexados:*** rendimiento baja cuando el archivo crece, bloques de desbordamiento (reorganizar el archivo).
- ***Ventajas de archivos de índice del árbol B+:*** se organiza automáticamente con pequeños cambios locales. No es necesario reorganizar todo el archivo.
- ***Inconvenientes de los arboles B+:*** inserciones extras, sobrecarga de borrados y costes de espacio.
- ***Ventajas de los árboles B+:*** superan inconvenientes, se usan ampliamente
- Satisface:
  - Caminos desde la raiz a la hoja, misma longitud
  - Cada nodo que no es raíz ni hoja tiene entre n/2 y n hijos.
  - Nodo hoja tiene (n-1)/2 y n-1 valores
  - Casos especiales: Si la raíz no es hoja, tiene al menos 2 hijos | Si la raíz es hoja, puede tener entre 0 y n-1 valores.

Estructuras de nodos del árbol B+

- **Nodo típico:** ki, valores de CB, Pi punteros a hijos (para nodos no hoja) o a los registros (para nodos hoja). CB ordenadas.

Nodos hoja

- Pi, apunta a un registro con valor de CB ki, o a un cajón de punteros a los registros de un archivo. Necesaria estructura de cajones si la CB no es primaria.
- El puntero apunta al siguiente nodo hoja, ordenado por clave de búsqueda.

Nodos que no son hojas: Forman índice disperso multinivel sobre hojas. Para un nodo hoja con m punteros:

- Todas las claves de búsqueda en el subárbol al que apunta P1 son menores que K1
- Para 2<=i<=n-1, todas las claves de búsqueda en el subárbol al que apunta Pi, tiene valores mayores o iguales que Ki-1 y menores que Km-1

Observaciones:

- Conexiones entre nodos con punteros, bloques lógicamente cercanos, no necesitan estar físicamente próximos.
- Número relativamente pequeño de niveles (log en el tamaño de archivo principal), búsquedas eficientes.
- Inserciones y borrados eficientes. El índice se puede reconstruir de forma log.

Asociación estática:

- Cajón. Unidad de almacenamiento con registros (bloque de disco)
- Organización de archivos asociativa, se obtiene el cajón de un registro con su valor de CB con una h.
- h. Función desde el conjunto de valores de CB k, hasta el conjunto de direcciones de cajones B.
- h se emplea para localizar registros para accesos, inserciones y borrados.
- Registros con diferentes valores de CB pueden asociarse a un cajón. Localizar registro, recorrer secuencialmente el cajón.

Funciones de asociación:

- La peor asocia todos los valores de CB al mismo cajón (tiempo proporcional al # de valores de CB del archivo)
- Ideal: uniforme, a cada cajón se asigna el mismo número de CB.
- Ideal: Ramdom, cada cajón mismo número de registros, independientemente de la distribución de valores de CB.
- Las h típicas hacen cálculos sobre la representación binaria de CB.

Gestión de desbordamiento de cajones: se puede producir por:

- Insuficientes cajones
  - Desviación en la distribución: Múltiples registros con mismo valor de CB | h produce distribución no uniforme
- Reducir probabilidad de desbordamiento, pero no eliminar, se gestiona empleando cajones de desbordamiento.
- ***Cadena de desbordamiento***: los cajones de desbordamiento de un cajón se encadenan en una lista enlazada (asociación cerrada). La asociación abierta, no emplea CD, no adecuada para apps de BD.

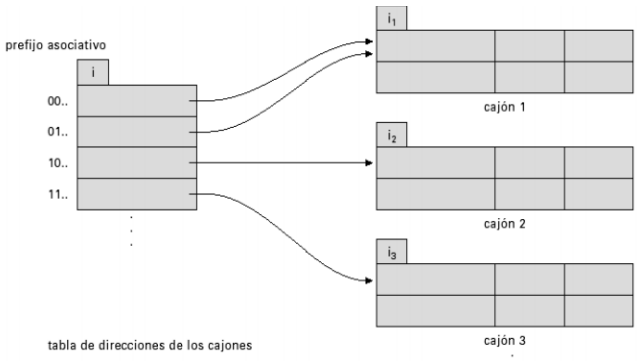
Deficiencias de la asociación estática: h asocia valores de CB a un conjunto B (tabla de direcciones de cajones).

- Las BD crecen. Si el # inicial de cajones es pequeño, disminuirá el rendimiento por desbordamientos.
- Si se anticipa el tamaño del archivo, el espacio se desperdiciaría inicialmente.
- Si disminuye la BD, se desperdicia espacio.
- Una opción es la reorganización periódica del archivo con otra h (costoso).

- Se evitan problemas, con técnicas que modifiquen el # de cajones dinámicamente.

Asociación dinámica:

- Buena para BD que aumentan y disminuyes de tamaño.
- Permite modificar dinámicamente la h
- ***Asociación extensible:*** asociación dinámica
  - h genera valores con un rango b-bit enteros, con b=32
  - Se usa solo prefijo de h, para indexar una tabla de B.
  - Sea la longitud del prefijo i bits, donde 0<=i<=32
  - Tamaño de B = 2^i, inicio i = 0
  - I crece y disminuye según el tamaño de la BD
  - Múltiples entradas en B pueden apuntar a un cajón.
  - Número real de cajones: <2^i (el # de cajones cambia dinámicamente por agrupaciones y divisiones).



Asociación extensible vs otros esquemas:

- Ventajas: prestaciones de asociación no disminuyen con el crecimiento del archivo. Costes espacio mínimos
- Inconvenientes: nivel extra de falta de dirección para encontrar registros. B puede ser muy grande: (estructura de árbol para localizar un registro). El cambio de tamaño de B es costoso.

Indexación ordenada vs asociación:

- Coste de reorganización periódica
- Frecuencia relativa de inserciones y borrados
- ¿Es deseable optimizar tiempo de acceso medio, a costa del tiempo de acceso del peor caso?
- Tipo esperado de consultas:
  - Asociación: mejor para recuperar registros con un valor determinado de clave
  - Si las consultas de rangos son comunes es mejor que los índices estén ordenados

Acceso multiclave:

- Emplear múltiples índices para ciertas consultas
- Se deben aplicar estrategias para procesar consultas empleando índices sobre atributos simples.
- Se pueden aplicar índices sobre múltiples atributos (con índices independientes es menos eficiente).

Índices de mapa de bits:

- Son un tipo especial de índice diseñado para consultas eficientes sobre claves múltiples.
- Registros en relación se enumeran secuencialmente. (debe ser fácil recuperar registros, si son de tamaño fijo)
- Aplicable sobre atributos/columnas que toman un # pequeño de valores distintos y con pocas actualizaciones
- Array/matriz de bits. En una coordenada todos los rowids de la tabla y en la otra los posibles valores del índice
- Un índice de mapa de bits tiene un mapa de bits por cada valor del atributo
  - El mapa de bits tiene tantos bits como registros | Si tiene valor 1, de lo contrario 0.
- Ventajas: Ahorro de espacio | Condiciones and/or en WHERE, rápidas
- Restricciones:
  - Utiles para and/or not in =
  - No adecuadas para comparación
  - No adecuadas para campos con muchas modificaciones

```
CREATE BITMAP INDEX emp_bitmap_idx
ON clientes (provincia);
```

- Tantas columnas como posibles valores del índice, tantas filas como registros en la tabla.
- Solo un “1” por fila de ROWID ya que ese campo en esa tupla solo puede tener un valor.