Daniel Ruskov Vangelov

DISEÑO DE BASES DE DATOS

VISTAS, RESTRICCIONES DE INTEGRIDAD Y DISPARADORES

Vistas

1. **Definición** - tabla virtual almacenada en el catálogo, definida en términos de tablas de base y sin almacenar tu extensión (tuplas). Una actualización sobre una vista se transforma en una actualización sobre las tablas de base y debe respetar las Restricciones de Integridad (RI). En caso de falsificarse alguna RI, la actualización sobre la vista da error. Se pueden definir vistas sobre vistas y una vista no se puede borrar si hay otras definidas sobre ella, a no ser que se use borrado en CASCADE de las vistas dependientes.
2. **Uso** - se utilizan para definir un esquema externo (parte de la BD que interesa a un grupo de usuarios ocultando el resto de la misma) y facilitar el acceso a consultas frecuentes y complejas. Facilitan la independencia lógica preservando los programas de cambios en el esquema conceptual y facilitar el mantenimiento de las funcionalidades y garantizan seguridad complementando los mecanismos de autorización (GRANT, REVOKE) definidos sobre las tablas.
3. **Actualización** - una actualización sobre una vista se transforma en una actualización sobre las tablas subyacentes. El borrado de una tupla de una vista puede propagarse al borrado de una tupla de la tabla base. Aparecen ciertos problemas:
   - 3.1. Ambigüedad - la actualización de una tupla en una vista puede llevar a la actualización de varios campos iguales en diferentes tablas. Estas no se dan si la vista está definida sobre una sola tabla o no contiene DISTINCT, funciones agregadas, atributos calculados, GROUP BY ni HAVING. También se puede incluir la opción CHECK, que garantiza que todas las actualizaciones verifican la condición de definición de la vista y puede verse como una RI más. Por último, la opción READ ONLY, que evita cualquier tipo de actualización sobre una vista.

Restricciones de Integridad

1. **Definición.**
   - 1.1. Integridad de los datos de una BD - es un estado de la BD en el cual los datos son correctos (no hay errores en la entrada de los datos, ni del programa de aplicación y no existe falsificación deliberada).
   - 1.2. Restricciones de Integridad - conjunto de predicados (reglas en un lenguaje de alto nivel) que deben verificar los datos de la BD. Dicho de otra forma, condiciones que se cumplen en el dominio. Aseguran que las modificaciones de una BD no dan lugar a pérdida de consistencia. Están compiladas y almacenadas en el catálogo. El ABD se encarga de su definición. Está el problema de su costosa comprobación. En UML se expresarán en OCL.
   - 1.3. Consistencia de una BD - un estado consistente es si se cumplen todas las RI.
2. **Comprobación -** las RI pueden ser violadas al operar con el SGBD. La aplicación y/o el SGBD pueden garantizar las RI
   - 2.1. Problemas:
     - 2.1.1. Repetición de código - para diferentes aplicaciones que usan la misma BD, o para asegurar la integridad de la BD cada aplicación tiene que comprobar las RI
     - 2.1.2. Mayor esfuerzo de mantenimiento - un cambio en las RI conlleva a cambios en las aplicaciones
     - 2.1.3. Mayor tráfico de datos en la red - datos de ejecución de la instrucción y datos de validación de la integridad.
   - 2.2. Ventajas arquitectura cliente-servidor:
     - 2.2.1. Descarga al servidor de datos.
     - 2.2.2. Comprobación inmediata para RI que no necesitan datos adicionales de la BD.
     - 2.2.3. Reutilización de código - código en un único sitio.
     - 2.2.4. Incremento de la consistencia - a todas las aplicaciones las mismas RI.
     - 2.2.5. Reduce el esfuerzo de mantenimiento - cambios en las RI conllevan cambiar solo el código declarativo en el SGBD.
     - 2.2.6. Eficiencia por reducción del tráfico en internet - sin tráfico de datos para verificar las RI.
   - 2.3. Conclusiones:
     - 2.3.1. Toda RI se verifica por el SGBD lo cual evita incoherencias, olvidos, etc.
     - 2.3.2. Adicionalmente se podrá hacer comprobación en el cliente con las RI que no impliquen datos adicionales.
3. **Lenguaje de Definición de Datos (LDD)** - definición de restricciones:
   - 3.1. Cláusula CONSTRAINT (SQL2) - especifica una RI en el contexto de una tabla y se viola si la condición devuelve FALSE al evaluarse (ej. ALTER TABLE CUADRO ADD CONSTRAINT suEst CHECK (estilo IN (‘moderno’, ‘clásico’));)
   - 3.2. Instrucción ASSERTION (SQL2) - especifica una RI: como unidad independiente y se viola si la condición se evalúa y devuelve FALSE.
   - 3.3. Disparadores TRIGGER (SQL3) - especifica una RI junto con acciones que se deben realizar para restaurar la integridad cuando se producen ciertos eventos.
4. **Tipos de Restricciones**

4.1. De atributo - condición sobre los valores que puede tomar cierto atributo de una tabla. Se viola si devuelve FALSE. No se viola si vale TRUE o UNKNOWN (caso de que un elemento valga NULL). Se comprueban al introducir o modificar una tupla, pero no al borrar.
  4.1.1. De clave primaria.
  4.1.2. De clave extranjera.
  4.1.3. De dominio.
    4.1.3.1. Dominios del sistema - VARCHAR2(30), INTEGER, DATE, TIME, etc.
    4.1.3.2. Dominios definidos por el usuario - DECLARE DOMAIN.
    4.1.3.3. Valores nulos - NULL, NOT NULL.
    4.1.3.4. Valores repetidos - NIQUE.
    4.1.3.5. Rango de valores - CHECK VALUE IN.

4.2. Intra-tupla - condición sobre los valores de los atributos de una tupla de una tabla. La condición puede referirse a otros atributos de la misma tupla (usando el nombre de los mismos). Se declara con la tabla. Se activa al introducir tupla en la tabla o actualizar tupla de la tabla, pero no al borrar tuplas de la tabla. No se garantiza si hay subpreguntas.

4.3. Inter-tupla - condición sobre varias tuplas de la misma o distintas tabla(s). Se declara como aserción. Los SGBD no suelen disponer de este mecanismo (solución de asociarlas a una de las tablas afectadas, definir disparadores). Se activa al introducir, actualizar y/o borrar cualquiera de las tablas de referencia. Está siempre garantizado.

5. **Estado de las restricciones:**

Nota: si hay varias RI a introducir se recomienda introducirlas todas como ENABLE NOVALIDATE, así el SGBD devuelve el control enseguida, y cambiar el estado a VALIDATE después, de forma que el SGBD lance varias transacciones en paralelo para comprobar la validez de los datos.

5.1. DISABLE - la RI no se comprueba.

5.2. ENABLE NOVALIDATE - la RI se comprueba sobre los nuevos datos, pero no sobre los ya existentes.

5.3. ENABLE VALIDATE (por defecto) - se comprueba sobre los datos existentes y los nuevos.

6. **Acoplamiento con las transacciones.**

6.1. Transacción - conjunto de operaciones sobre la BD que forman unidad atómica de trabajo con el SGBD, o dicho de otra forma, forman una unidad de integridad, consistencia. De parte de un estado donde se verifican las RI, durante la ejecucion se pueden violar las RI, y se debe llegar a un estado que verifica las RI. La comprobación se puede hacer durante la transacción (al realizar la operación) mediante acoplamiento inmediato, o al finalizar la transacción mediante acoplamiento diferido.

6.2. IMMEDIATE - se comprueban las restricciones nada más producirse la modificación. Detecta inmediatamente la violacion de RI y acorta el proceso de deshacer. No se puede usar si existen estados intermedios inconsistentes.

6.3. DEFERRED - se comprueban las restricciones antes de finalizar la transacción. Justo antes del COMMIT. Si detecta violacion de RI, hay que deshacer toda la transacción. Se puede usar con estados intermedios inconsistentes, ya que así permite que una operación reponga la validez de una RI que haya sido falsificada por una operación anterior.
  6.3.1. NOT DEFERRABLE - en transacciones posteriores no se puede usar la cláusula SET CONSTRAINT para poner en opción DEFERRED.
  6.3.2. DEFERRABLE - en transacciones posteriores se puede usar la cláusula SET CONSTRAINT para poner en opción DEFERRED. Se debe especificar el modo de acoplamiento inicial:
    6.3.2.1. INITIALLY DEFERRED - la validación después del COMMIT.
    6.3.2.2. INITIALLY IMMEDIATE -la validación ocurre en el momento.
    6.3.2.3. Por defecto - NOT DEFERRABLE INITIALLY IMMEDIATE

7. **Mantenimiento** - hay que decidir qué hacer cuando una operación falsifica una RI:

7.1. Impedir la operación.

7.2. Permitir la operación - recuperando la validez de la RI haciendo cambios adicionales (reglas de compensación),

Disparadores

1. **Motivación** - dinamicidad de toma de acciones ante las RI de forma explícita (evento y acción según condición). Traslado de la lógica de negocio/aplicación al SGBD.
2. **Descripción -** Reglas ECA (Evento + Condición + Acción).

2.1. Evento - suceso que causa que se desencadene un disparador (INSERT, UPDATE, DELETE, CREATE, ALTER, DROP, LOGON, LOGOFF, STARTUP, SHUTDOWN, SERVERERROR)

2.2. Condición - expresión booleana que debe ser TRUE para que se ejecute el disparador. No se ejecuta si se evalúa a FALSE o UNKNOWN.

2.3. Acción - procedimiento que se debe ejecutar cuando se dispara el disparador.

2.4. Clasificación de las reglas ECA:
  2.4.1. Por la acción que desencadena:

2.4.1.1. Disparadores de tupla (row triggers) - modificación de una tupla. Se desencadena el disparador una vez por cada una de las tuplas afectadas (borradas, introducidas, actualizadas) por la sentencia que desencadena el evento.
2.4.1.1.1. OLD - variable que contiene la tupla antes de la modificación. No tiene sentido si el evento es INSERT.
2.4.1.1.2. NEW - variable que contiene la tupla después de la modificación. No tiene sentido si el evento es DELETE.
2.4.1.1.3. En el cuerpo del trigger, NEW y OLD deben ir precedidas de ":", pero no en la cláusula WHEN.
2.4.1.1.4. La cláusula REFERENCING se usa para asignar alias a las variables NEW y OLD.
2.4.1.2. Disparadores de sentencia (statement triggers) - ejecución de una sentencia. Se desencadena el disparador una sola vez (sin tener en cuenta las tuplas afectadas) por toda la tabla.
2.4.1.2.1. OLD_TABLE - conjunto de tuplas afectadas antes de la modificación.
2.4.1.2.2. NEW_TABLE - conjunto de tuplas afectadas después de la modificación.
2.4.1.2.3. Oracle no soporta OLD_TABLE y NEW_TABLE.

2.4.2. Por el momento en el que se desencadena la regla con respecto a la operación sobre la BD:
2.4.2.1. Disparadores before (BEFORE trigger) - se ejecuta antes de que se realice la operación que desencadena el evento. Solo para tablas de base y no para vistas. Si se inserta en una vista y la tabla de base subyacente tiene un disparador BEFORE, entonces sí se dispara.
2.4.2.2. Disparadores after (AFTER trigger) - se ejecuta después de que se realice la operación que desencadena el evento. Solo para tablas de base y no para vistas. Si se inserta en una vista y la tabla de base subyacente tiene un disparador AFTER, entonces sí se dispara.
2.4.2.3. Disparadores instead of (INSTEAD OF trigger) - se ejecuta en lugar de la operación que desencadena el evento. Sirve para vistas. Es una manera transparente de modificar vistas no modificables. Si se sabe cuál es la operación asociada a una inserción en una tupla, se puede indicar en un disparador la operación asociada.

3. **Estado de un disparador:**
3.1. ENABLE (por defecto) - activo, se dispara.
3.2. DISABLE - inactivo, no se dispara.

4. **Acoplamiento:**
4.1. IMMEDIATE - la condición se evalúa dentro de la misma transacción que la del evento que la desencadena y se hace inmediatamente. Uso en condiciones que requieren una comprobación inmediata.
4.1.1. BEFORE - se evalúa la condición antes de la ejecución de la operación.
4.1.2. AFTER - se evalúa la condición después.
4.1.3. INSTEAD OF - se evalúa la condición en lugar de realizar la operación.
4.2. DEFERRED - la condición se evalúa al final de la transacción que contenía la operación desencadenadora. En este caso puede haber varios disparadores esperando a evaluar su condición. Uso en condiciones complejas diferidas.
4.3. DETACHED (separado) - la condición se evalúa dentro de una transacción separada nacida de la transacción desencadenadora. Uso en condiciones complejas diferidas en otra transacción

5. **Uso.**
5.1. Restauración de la integridad.
5.2. Mantener datos derivados v.s. atributos derivados.
5.3. Mantener réplicas síncronas de tablas.
5.4. Mantenimiento de vistas materializadas.
5.5. Auditar ciertos eventos aparte de los que ofrece el SGBD (Crear un "log" con eventos que nos interesen y recoger estadísticas de actualización a las tablas).
5.6. Llamar a sistemas externos al SGBD (Otras aplicaciones asociadas al SGBD, informar a otras aplicaciones sobre eventos sobre la BD).
5.7. Notificar situaciones de los datos a los usuarios (enviar e-mail, mensaje al móvil, busca).

6. **Ventajas:**
6.1. Promueve la reutilización del código.
6.2. Incrementa la consistencia.
6.3. Reduce el esfuerzo de mantenimiento.
6.4. Reduce el tráfico por la red.

7. **Inconvenientes:**
7.1. Difíciles de depurar si el número es grande - posibles efectos laterales insospechados.
7.2. Puede sobrecargar al sistema dependiendo de su número y la complejidad de la condición.
7.3. No se garantiza la terminación del proceso de disparo (efecto cascada, un trigger puede llamar a otro trigger, situación deadlock, etc). No hay herramientas comerciales disponibles. SQL Server + Oracle (máximo 32 anidamientos).
7.4. Inexistencia de metodologías claras.

DISEÑO FÍSICO

Objetivo

1. **Se distinguen dos categorías en la organización de ficheros:**
   - 1.1. Métodos de organización primarios - relacionados con las estructuras de almacenamiento (organización física del fichero, orden físico de los registros) . Se sustentan en un atributo (o conjunto de atributos) concreto. Existe sólo una organización primaria de un fichero.
   - 1.2. Métodos de organización secundarios - relacionados con las estructuras de acceso (tipos de búsqueda) para recuperar eficientemente registros de acuerdo a valores de atributos que no fueron considerados en las organizaciones primarias. No influyen en el lugar donde están almacenados los registros y usan lo que se denomina índices secundarios.
2. **Criterios a utilizar** - normalmente se consideran el caso medio y el peor caso bajo diferentes decisiones de diseño. Se utilizan técnicas de simulación.
   - 2.1. Tiempo de respuesta - tiempo comprendido desde que se envía la transacción hasta que se recibe la respuesta.
   - 2.2. Espacio utilizado.
   - 2.3. Throughput - productividad de las transacciones.
3. **Tipos de almacenamiento** - la mayoría de las BD se almacenan de forma permanente en almacenamiento secundario. Los datos almacenados en el disco se organizan como ficheros de registros.
   - 3.1. Principal o primario - memoria principal (almacenamiento volátil).
   - 3.2. Almacenamiento secundario y terciario - discos y cintas (almacenamiento no volátil).
4. **Almacenamiento y búsqueda:**
   - 4.1. Algunas organizaciones de ficheros admiten usar varios tipos de búsqueda.
   - 4.2. Interesa utilizar la organización de ficheros más conveniente. Será la que permita realizar de manera más eficiente las operaciones más frecuentes.
   - 4.3. En muchos casos ninguna organización permite que todas las operaciones frecuentes se implementen eficientemente. Entonces se toma una solución de compromiso.

Estructuras de almacenamiento

1. **Ficheros desordenados, o de montón (heap)** - registros no ordenados. Es la organización más simple.
   - 1.1. Búsqueda - lineal. Bloque a bloque para encontrar todos los registros que cumplan la condición. Costoso.
     - 1.1.1. Mejor caso de un bloque cuando en el primer bloque se encuentra un registro que cumple la condición y la condición incluye una clave.
     - 1.1.2. Peor caso de varios bloques cuando ningún registro cumple la condición y la condición no incluye ninguna clave.
   - 1.2. Inserción - se pone en el búfer el último bloque del fichero, si hay espacio en él se añade el registro y si no lo hay se crea bloque nuevo en el búfer. Se sobrescribe o escribe el bloque del buffer en el disco. En su caso, se guarda la dirección del bloque nuevo en el descriptor.
   - 1.3. Eliminación - se busca en el disco el registro a borrar, después se transfiere el bloque que lo contiene al búfer de memoria y se elimina. Otra opción es usar un bit extra por cada registro como marcador de eliminación. Se sobreescribe el bloque del disco.
     - 1.3.1. Problema de desperdicio de espacio - deja espacios vacíos en el disco.
       - 1.3.1.1. Solución 1 - organización periódica del fichero recorriendo en secuencia todos los bloques y juntando los registros eliminando espacios.
       - 1.3.1.2. Solución 2 - buscar primero un hueco para insertar y si no lo hay hacerlo al final. Supone búsqueda y consumo de tiempo.
   - 1.4. Modificación - se busca el registro a modificar, se transfiere su bloque al buffer, se modifica y se escribe de nuevo sobre el bloque del disco.
     - 1.4.1. Problema de desbordamiento del bloque - con registros de tamaño variable que han crecido al modificarlos.
       - 1.4.1.1. Solución - eliminar del fichero el registro a modificar e insertar en el fichero el registro modificado.
   - 1.5. Recomendación - usar cuando los datos tienen que ser cargados en la tabla en grandes cantidades, cuando la tabla sólo ocupa unas pocas páginas, el tiempo para localizar un registro será corto y cuando cada vez que se accede a la tabla, tienen que ser recuperadas todas las tuplas, pero no cuando de la tabla sólo se recuperan tuplas seleccionadas.
2. **Ficheros ordenados, o secuenciales** - sus registros se mantienen físicamente ordenados de forma automática según los valores de un campo especial (campo de ordenación) que puede ser la combinación de varios campos. Si ese campo es clave, se llama clave de ordenación. Los ficheros ordenados no se usan apenas tal y como se presentan aquí. Sin embargo

su uso es muy habitual acompañado de un índice. Esto se conoce como índice primario (para campos clave de ordenación, por ejemplo, para claves primarias de tablas) e índice de agrupación (cuando el campo de ordenación no es clave).

2.1. Ventajas:
  2.1.1. Obtiene de manera eficiente los registros en el orden del campo de ordenación.
  2.1.2. Proporciona acceso eficiente al siguiente registro en el orden del campo de ordenación (estará en el búfer si quedan registros por leer, si no, trae el siguiente bloque si hay)

2.2. Desventajas:
  2.2.1. Como se debe mantener continuamente el orden de los registros por el campo de ordenación, la inserción y eliminación son operaciones costosas. Alternativa de bloques con espacio libre y cuando se llenan tenemos el mismo problema o área de desbordamiento (fichero de desbordamiento) donde los registros no están ordenados, insertan al final y periódicamente se reconstruye el fichero ordenado, colocando en su posición los registros del área de desbordamiento. Así la inserción es más eficiente, pero la búsqueda es algo más ineficiente.

2.3. Búsqueda - binaria (dicotómica), mucho más rápida, sólo si la condición de búsqueda es sobre un valor del campo de ordenación. Se efectúa sobre bloques. Normalmente se accede a log2(b) bloques. También permite buscar por <, >, <=, >= sobre el campo de ordenación.

2.4. Inserción - Se busca el bloque donde se debe insertar con búsqueda binaria, se transfiere al buffer, se abre espacio en el bloque del buffer para el registro desplazando al resto, se inserta el nuevo registro y se sobreescribe en disco. Si no había espacio en el bloque habrá que desplazar el registro sobrante (con valor mayor del campo de ordenación) al bloque siguiente. En el caso peor esto afecta a todos los bloques del fichero.

2.5. Eliminación - similar a la inserción para la eliminación física. Resulta más sencillo si sólo se marca como borrado y se reorganiza periódicamente.

2.6. Modificación - según el caso podrá realizarse búsqueda binaria. Si se modifica el campo de ordenación puede precisar cambiar de posición el registro. Eso supone hacer una eliminación y una inserción. Si los registros son de longitud fija, el bloque podrá reescribirse en la misma posición.

3. **Ficheros de direccionamiento calculado (hashing)** - proporciona el acceso más rápido para ciertas búsquedas (en general se transfiere un solo bloque). Los ficheros de direccionamiento calculado se llaman también ficheros dispersos o directos. Condición de búsqueda es la igualdad sobre el campo de direccionamiento calculado(c.dir.cal.), que en la mayor parte de los casos es clave. Se usa función hash, aplicándola al valor del c.dir.cal.del registro y se obtiene la posición de una tabla que tiene una dirección de bloque de disco que contiene el registro con ese valor de c.dir.cal. Se denomina direccionamiento calculado externo o hashing.

3.1. Búsqueda - por campo diferente a c. dir. cal. será lineal.

3.2. Eliminación - si está en el cubo principal, pasar un registro del área de desbordamiento. Si en el cubo de desborde, eliminación en la lista enlazada.

3.3. Modificación - si es c. dir. cal. se elimina y vuelve a insertar, si es sobre otro campo se reescribe.

3.4. Problemas:
  3.4.1. Las funciones hash, en general, no preservan el orden de la clave.
  3.4.2. Cantidad de espacio prefijada (desperdicio de espacio o desbordamientos).
  3.4.3. Solución - expansión dinámica.
    3.4.3.1. Direccionamiento calculado extensible mediante un directorio más cubos.
    3.4.3.2. Direccionamiento calculado lineal con varias funciones hash (se aplica una función, cuando se produce una colisión se aplica otra función y así sucesivamente).

3.5. Recomendación - usar cuando las tuplas se recuperan en base a la coincidencia exacta (igualdad) con el valor del campo de hash, pero no cuando las tuplas se recuperan en base a un rango de valores del campo de hash, ni cuando las tuplas se recuperan en base a un atributo que no es el del campo de hash, ni cuando las tuplas se recuperan en base a una parte del campo de hash.

Estructuras de acceso

1. **Índices** - almacena valores del atributo junto con los punteros. Los valores del índice están ordenados. Un índice denso tiene una entrada por cada registro. Un índice no denso (disperso) tiene una entrada por cada bloque. Aceleran la búsqueda de registros. Se puede construir un índice sobre un campo del fichero o sobre un conjunto de campos del fichero y también pueden existir índices multinivel (un índice sobre otro). La mayoría implementados mediante árboles B^+(balanceados). Tipos:

1.1. Primarios sobre clave (IP) - exige que la organización primaria sea fichero ordenado por campo de indexación. Solo puede existir un índice primario. Se debe a que los registros de un fichero sólo pueden mantenerse ordenados físicamente por un único concepto.
  1.1.1. Ordenamiento lógico - aquí no tiene sentido ya que los registros están ordenados físicamente.
  1.1.2. Búsqueda de un registro - igual que en IS-c. Toda búsqueda visita un nodo de cada nivel y el bloque que tiene el registro con el valor buscado (único, ya que es clave)

1.1.3. Inserción/eliminación de registros- -como con cualquier índice, supone actualizar y reorganizar el índice

1.2. Secundarios sobre clave (IS-c) - admite cualquier organización primaria. Sin ordenación física.

1.3. Secundarios sobre no clave (IS-nc) - admite cualquier organización primaria. La diferencia con IS-c es que las hojas pueden apuntar a los varios registros con idéntico valor de campo de indexación. Por eso incluye un nivel de indirección a bloques de punteros(existen otras alternativas a este nivel de indirección).

1.3.1. Ordenamiento lógico - poder leer los registros en orden del campo(s) de indexación. Sólo para índices secundarios (los primarios y de agrupación tienen otra opción mejor).

1.3.2. Búsqueda de un registro - igual que IS-c, salvo que aquí hay que visitar el nivel con bloques de punteros. Toda búsqueda visita un nodo de cada nivel, un bloque de punteros (o varios, si no caben todos los punteros en un bloque) y uno por cada bloque que tenga registros con el valor buscado (puede haber varios, ya que no es clave).

1.3.3. Inserción/eliminación de registros - como con cualquier índice, supone actualizar y reorganizar el índice

1.4. De agrupación sobre no clave (IA) - exige que la organización primaria sea fichero ordenado. Solo puede existir un índice de agrupación. Se puede construir similar al IS-c, pero cada valor de las hojas apunta al bloque de datos donde está el 1º registro con ese valor.

1.4.1. Ordenamiento lógico - tampoco tiene sentido ya que los registros están ordenados físicamente.

1.4.2. Búsqueda de un registro - el primer registro con ese valor se busca igual que en IS-c. El resto de registros con el mismo valor están seguidos al 1º en el fichero de datos. Toda búsqueda visita un nodo de cada nivel y los bloques de datos que tienen registros con el valor buscado (pueden ser varios, ya que no es clave).

1.4.3. Inserción/eliminación de registros - como con cualquier índice, supone actualizar y reorganizar el índice.

2. **Árboles B^+** - siempre equilibrado, conlleva un desperdicio de espacio aceptable. Inserción y eliminación de cierta complejidad. Los nodos internos y las hojas son diferentes. En nodos hoja hay valores y apuntadores a bloques de datos.
3. **Índices sobre clave múltiple** - que se define sobre un conjunto de campos. Además de los índices de agrupación, también se pueden definir índices primarios y secundarios sobre clave compuesta.
4. **Direccionamiento partido.**
5. **Ficheros rejilla.**

Recomendaciones de diseño diap 44 →
Tuning de consultas diap 62 →

PEREZA DE RESUMIR MÁS