uc3m | Universidad Carlos III de Madrid

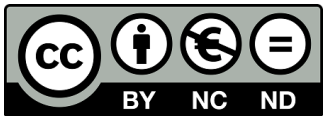
Grado en Ingeniería Informática

2019-2020

Apuntes

Ficheros y bases de datos

Jorge Rodríguez Fraile[1]



[1]Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

ÍNDICE GENERAL

1. TEMA 1: INTRODUCCIÓN A LAS BB. DD

1. **Informática:** Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores.

   Información: Concepto más abstracto de dato. Comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada.

   Dato: Información dispuesta de manera adecuada para su tratamiento por un ordenador.

Transmitir información:

1. Requiere que los sujetos compartan la misma codificación. Los sucesos físicos pueden ser persistentes, que permanecen en el tiempo por lo que permite almacenar información, o inestables, que son volátiles y desaparecen con el tiempo.

   Características:

   a) **Perdurabilidad:** la información dura poco o mucho tiempo.
   Capacidad: Cantidad de información, relativa al coste o al espacio.
   Velocidad: Tiempo necesario para acceder a la información.
   Alcance: La información es accesible por uno o más receptores.
   Tipo de acceso: Privilegiado o externo.

   Soporte principal(RAM/ Escritorio): PROCESAR

   a) Ágil, acceso rápido y poca información por acceso.
   Más costoso y requiere más espacio.
   Poco alcance, solo el usuario puede acceder a el.
   Volátil.

   Soporte secundario(Disco/ Estantería): ALMACENAR

   a) Lento, accesos externos y mucha información por acceso.
   Menores costes y espacio.
   Gran alcance, muchos usuarios pueden acceder.
   Persistente.

Soporte de almacenamiento: Material capaz de registrar información.

Dispositivo de almacenamiento: Soporte (Hardware) capaz de proporcionar lo necesario para el almacenamiento y la recuperación, escribir y leer.

Fichero: Cada unidad contenedora de información en el soporte. Pueden ser subdivisiones de algo más grande o el original.

a) Nominados, deben estar identificados.
Estructurados de forma útil.
En un soporte no volátil.

Archivo: Cada unidad contenedora de información para los usuarios.

El enfoque lógico o externo (lo hace el usuario), busca la EFICACIA:

a) Añadir (Insertar), Recuperar(Consultar), Editar(Modificar), Eliminar(borrar) o Buscar(Seleccionar)

El enfoque físico o interno (lo hace la máquina), busca la EFICIENCIA, menores costes de recursos:

a) Leer, Escribir y en las más avanzadas Localizar. Son las operaciones simples que permiten realizar las de lógico.

Estructuras Físicas:

1. ¿Qué pueden hacer?

   a) Insertar, añadir nuevos elementos.
   Eliminar, quitar un elemento.
   Modificar, hacer cambios/ sustituir.
   Consultar.

   Organización Serial: La ausencia de organización (Amontonado).

   a) Inserción óptima, no necesitar localizar, se deja donde sea.
   Ahorro espacio, no deja espacios.
   Full Scan más barato, al estar todo junto.
   Localización pesada, tengo que mirar «todo» para buscar un elemento.

Organización Secuencial: Ordenados siguiendo un criterio específico, en fila.

a) Más fácil encontrar un elemento, si es según el criterio, si no tendrá que recorrer todo, por eso la indizada es mejor por si queremos cambiar el criterio de búsqueda.

Necesita mantenimiento, hacer hueco cuando haya que insertar un elemento en medios, degenera y dificultará el mantenimiento.

Organización Direccionada: Ordenados por dispersión, hash, conociendo un dato del elemento sabemos localizarlo más fácilmente. (Ejem. Las clases y su codificación).

a) Selección óptima para esa clave, pero para el resto de claves empeora.

Desperdicia mucho espacio, por ello para leerlos todos tardo más.

Organización Indizada: Existen un lugar de consulta, indices que son estructuras auxiliares, que mejoran el tiempo de acceso.

a) Puede seguir varios criterios y el coste de selección es reducido.

Estructuras lógicas vs. Estructuras físicas:

1. 1 a 1: Cada archivo en un fichero.

   n a 1: n archivos en un fichero.

   1 a n: 1 archivo que ocupa n ficheros

   n a n: n archivos por fichero que ocupa en total n ficheros.

Arquitectura ANSI/SPARC:

1. Enmarcas las estructuras de las bases de datos, en tres niveles:

   a) **Nivel interno:** Como se relacionan los datos con el soporte.

   Nivel conceptual: Como se relacionan los datos con los datos. Sin tener en cuenta para qué se van a usar esos datos, ni como van a ser físicamente almacenados.

   Nivel externo: Visión de la base según cada tipo usuario, como se relacionan los datos con los usuarios.

Modelo de Datos: Pasará a ser una estructura de datos (Grafo, diagrama) y la idea es obtener las propiedades y restricciones del universo de discurso.

1. Restricciones: Limitaciones impuestas sobre la Base de Datos.

   a) **Restricciones inherentes:** Son las propias de la herramienta, impuestas sobre la estructura del modelo.
   Restricciones semánticas: Son las propias del problema, impuestas sobre los datos.

   Propiedades estáticas: Invariantes en el tiempo, que permiten describir estructuras. Objetos, asociaciones y restricciones. Pueden estar asociados, que también serán información y aportan restricciones.

   Propiedades dinámicas: Variante en el tiempo, que permiten describir operadores.

Concepto de Base de Datos: Colección o depósito de datos integrados, con redundancia controlada (al ser controlada no es mala, si hay un mecanismo para controlarla),

con una estructura que refleja las interrelaciones y restricciones del mundo real (buscar resolver un problema que no existe, resolver el que tiene el cliente),

cuyos datos serán independientes de aplicación o usuario (tiene que servir para cualquier uso, porque tiene que ser de esa manera, para que sirva para un futuro no solo para ese caso)

y tendrán definición y descripción única (cada dato tendrá unos determinados atributos y no más, y los datos sobre los datos son los metadatos, se almacenan junto a los datos),

y cuyos procedimientos involucrados preservarán la integridad de la Base (si desaparece/-borramos un dato, que la base no deje de ser útil),

respetando además ciertas normas de disponibilidad y confidencialidad(solo los que tienen autorización pueden acceder y lo puedan hacer en cualquier momento, pero nadie más que no tenga la autorización. La Seguridad).

10. **Sistema Gestor de Bases de Datos:** Conjunto de herramientas (Programas, procedimientos, lenguajes, ...) capaz de posibilitar la interacción (describir, recuperar y manipular) con la base de datos a todos lo niveles(usuarios de todo tipo, programador, analista, diseñador, ... y administrador, que controla la estructuras físicas).

    a) Funciones esenciales: Que se resumen en 3 lenguajes esenciales.

        1) **Lenguaje de Descripción:** Permitir definir los elemento de datos y sus estructuras.
        Lenguaje de Manipulación: Posibilitar la operación del contenido de la base.
        Lenguaje de Utilización: Conjunto de herramientas para que el administrador pueda desarr

2. TEMA 2: ESTÁTICA RELACIONAL

Los modelos relacionales se basa en la noción matemática de relación, R: N x N**:**

- Se pueden describir mediante una función, R={(x, y) / $y=x^2$ }
- O se pueden enumerar las relaciones, R={(1,1), (2,4),...}

Dominio: Conjunto de valores de la misma naturaleza. Puede tener restricciones.

Relación: Subconjunto de productos cartesianos, se pueden dar entre elementos del mismo dominio o entre elementos de distinto dominios.

- Nombre x Nombre. Persona: Nombre x Edad x Altura x Teléfono

Atributo: Propiedad común entre los elementos de una relación, y se define sobre un dominio. Ej. edad se define sobre el dominio de los enteros entre 0 y 120 años.

- **Definición Intensiva:** Definición invariable, no cambia con el tiempo. Se representa como: etiqueta(atrib1, atrib2,...)
- **Definición Extensiva**: Definición variable, para referirnos uno de los miembros en un momento del tiempo. Se representa en una tabla, cada individuo en una fila y como cabecera estaría la definición intensiva y el nombre del atributo.

Esquema de una relación: Asociación de atributos que caracteriza y distingue a los miembros de una relación. relación —> etiqueta(atrib1, atrib2, atrib3,...)

- Persona (Nombre, DNI, Teléfono)
  - **Grado**: Numero de atributos, es estático.
  - **Cardinalidad**: Numero de tuplas, numero de filas en una tabla, dinámico.

Restricciones Inherentes:

- El orden de las tuplas no es significativo (filas), ni tampoco el de los atributos dentro de la tupla(columnas). Lo importante es que definido el orden, todos los atributos los almacenemos en el mismo orden.
- No hay dos tuplas iguales, no completamente. Deben identificar individuos.
- Cada atributo toma un solo valor del dominio dentro de una tupla.

- **Integridad de Entidad:** Si algo existe es identificable, si no es identificable no existe. Y ese identificador no puede tomar valor nulo.
- **Integridad Referencial:** Todo valor referido o lo referenciado existe, debe apuntar al dato y no puede dejar de hacerlo o se rompe la integridad.

Representación y notación : Se usa camelCase y snake_case, preferiblemente la última. Los nombres deben ser representativos y no se pueden repetir en el mismo ámbito las etiquetas.

- **Definición intensiva** (es la que utilizaremos): grafo relacional.
- **Definición extensiva**: representación tabular(en una tabla).
- Los dominios tienen etiqueta en plural. Ej. Personas, Teléfonos...
- Los atributos tienen etiqueta en singular(ya que solo hay un valor). Ej. DNI, nombre, teléfono...

Ocurrencia o Tupla: Asociación de valores para un individuo. <val1, val2, val3,...>

Atributos opcionales: No tienen por qué tener valor, cuando esto ocurre toma el valor null, 0 o ‘ ’. El dominio incluye el nulo. Se indica con un *. Esto se da cuando:

- Es desconocido por el usuario. No la conoce todavía.
- No aplicable. No siempre se tiene por los que puede dejarse en blanco.
- Es desconocido por mi. No se ha sido dado al sistema.

Atributo obligatorio: Cuando no se acepta el valor nulo, no se puede dejar en blanco, el dominio no contiene al nulo.

Clave: Atributo o conjunto de atributos con función definida. Ej. nombre, para ordenar alfabéticamente.

Superclave: Atributo que es capaz de identificar aún individuo. Siempre hay una, aunque no sea mínima, ya que las tuplas son únicas. No se mantiene para todos los tipos de relaciones. Lo seguirá siendo aunque añadamos más atributos, pero no será mínima. Es mínima cuando al eliminar cualquier atributo deja de ser superclave. NO puede ser un atributo opcional.

Clave ajena: Atributo o conjunto de atributos de una relación referenciante utilizados para apuntar o vincular cada fila con una fila de otra relación, la relación referenciada. El atributo referenciante, apunta a una superclave, se indica con una flecha desde la referenciante (puede ser opcional) a la referenciada(obligatoria). Si es múltiple se puede

desdoblar para no equivocarse. Si la referenciante apunta a un valor este debe existir para mantener la integridad.

Tipo de relaciones entre esquemas:

- **1 a 1:** Correspondencia biunívoca. La clave ajena puede localizarse en cualquiera de los dos. Fuerte—>Débil. La más débil será la más volátil o por existencia. Se pondría fusionar ambos esquemas en uno solo.
- **1 a n:** Correspondencia múltiple(1 del primero se relaciona con varias del segundo), los hijos son los que contienen la clave del padre. Los hijos apuntan al padre. La clave ajena está en la relación con múltiples tuplas (llave)
- **m a n:** Surge de un problema de diseño, se resuelve añadiendo una nueva relación, una relación intermedia. Y los atributos de la relación intermedia que han llevado a su creación son superclaves de la relación. dl_club_persona(..., ..., ...)

Clave candidata: Superclave mínima, si eliminamos una atributo cambia el orden de los elementos, se elige la más corta o que más aparece como primaria.

Restricciones Semánticas:

- **de Rechazo:** Rechaza las operaciones sobre los datos que rompen la restricción semántica o no contempladas.
  - Simple: Solo afecta a un elemento relacional, dominio, relación o tupla.
  - Aserción: Afectan a varias tablas, filas, ... aunque no esté operando en ella. En la práctica no es viable, es teórico, en la práctica se usa disparadores.
- **Clave primaria**: Clave candidata privilegiada. Se indica subrayando con una línea continua. NO puede ser opcional.
- **Clave alternativa**:El resto de claves candidatas. Se indica subrayando con una línea discontinua. NO puede ser opcional. Si son varios atributos con una llave.
- **Obligatoriedad (NOT NULL)**

Reglas de Integridad Referencial: Corrección automática para evitar que se pierda la integridad referencial.

- **Restrict(R):** No se lleva a cabo si tiene hijos, aborta la operación y el usuario decide que hacer.
- **No Action(NA):** Se borra y después comprueba si tiene hijos, si los tiene se restaura el padre, pero si no los tiene se deja borrado.

- **Cascade(C):** Si borro o modifico el padre, le ocurrirá lo mismo a los hijos.
- **Set Null(SN):** Si es opcional el atributo, cuando se borra el padre los hijos toman el valor nulo.
- **Set Default(SD):** Se pone el valor por defecto en los hijos cuando se borra el padre.

Fases del Diseño relacional: En las 3 primeras debe estar toda la semántica.

- Diseño, hacer el grafo.
- Implementación, hacer las tablas.
- Incorporación semántica, hacer las restricciones mediante disparadores.
- Elementos externos.

Documentación: Sirve para completar el grafo con comentarios acerca de la cobertura semántica de la solución propuesta.

- Los supuestos semánticos explícitos son los requisitos aportados por el cliente/usuario.
- **Los supuestos semánticos explícitos no contemplados:** Me lo han pedido y no ha sido posible hacerlos. Se enumeran y explican casos que no han sido posible implementarlos en el diseño propuesto, y se puede proponer una solución para ellos.
- **Los supuestos semánticos implícitos**: No me lo han especificado y lo he hecho. Cuando el diseño necesita más información que la proporcionada por el cliente, y se implementa en el diseño y afecta a la cobertura. Hay que indicar todo lo que añado y restringe el diseño.

Dependencia funcional x—>y: y es dependiente de x, si dado el valor del antecedente(x) el consecuente(y) es inequívoco. Ejem: DNI—>nombre.

Forma Normal: Sirve para validar que el diseño es correcto, evitando redundancias o defectos, cada vez son más restrictivas. Estudiaremos las 4 principales, el resto se usan muy poco.

- **1NF:**
  - Todos los atributos son atómicos (indivisibles).
  - Los atributos tienen un solo valor, monovaluados.
  - Tiene clave primaria, superclave mínima no nula.

    - Todas las tuplas con la misma cantidad de atributos, grado fijo, los puede haber nulos.
    - No importa el orden de las filas y columnas.

- **2NF:** Está en 1NF y no existen dependencias parciales de una clave candidata. Un subconjunto de la clave primaria tiene una relación funcional sobre algún otro atributo no primo, que no pertenece a ninguna clave. Se resuelve creando una nueva relación.
- **3NF:** Está en 2NF y no existen dependencias transitivas entre la clave y no primos, que no pertenece a ninguna clave. Que haya una dependencia de un atributo que no es primo con otro que tampoco lo es. Se resuelve con una nueva relación que recoja ambos atributos.
- **BCNF:** Está en 2FN y no existen dependencias transitivas. No restringe a que no sean primos. No todas las 3NF están en BCNF.

3. TEMA 3: DINÁMICA RELACIONAL

Operar con la base de datos. Las operaciones sobre la BD producen nuevos estados en la BD.

Los lenguajes relacionales son de especificación. Se distinguen:

- **Procedimentales o Algebraicos**: Especifica los pasos, hasta llegar a la meta, mediante operaciones que sufre la BD.
- **No-procedimentales**: Indica la meta, pero no como la alcanzaremos, como serie el estado final de la BD.

Los operandos en cualquier operación algebraica-relacional son relaciones, y el resultado es siempre una relación. No se modifican las originales, se crea otra nueva tabla.

Operadores unarios AR Básica: Recibe una relación y devuelve otra relación.

- **Selección**: Mismas columnas, pero solo las filas que cumplan la condición. Se designa con un sigma. grado original=grado final y cardinalidad original>=cardinalidad final.
- **Proyección**: Tiene menos columnas siempre, pero las mismas filas, se indican los atributos que formaran la nueva relación. grado original > grado final y cardinalidad original= cardinalidad fin.
- **Renombrado**: Le doy una etiqueta a una operación, para ahorrar repetir la expresión.

Operadores binarios AR Básica: Para las 3 primeras las relaciones deben ser compatibles, mismo grado y definición de los atributos. Recibe dos relaciones compatibles y devuelve otra relación.

- **Unión**: Tiene todas las filas que tenían ambos operandos. grado 1=grado 2=grado final y cardinalidad final <= cardinalidad1 + cardinalidad2
- **Intersección**: Las filas comunes en ambas tablas. grado A=grado B= grado final y cardinalidad final <= cardinalidad del que menos filas tiene.
  - A n B=A-(A-B)
- **Diferencia**: Todas las filas que están en A, pero que no estén también en B. Grados iguales y cardinalidad final <= cardinalidad de la relación de la izquierda.

- **Producto cartesiano**: Tiene todos los atributos de A y B, y en contenido todas las filas de A concatenadas con las filas de B, todas las posibilidades de juntar ambas filas. grado final=grado A+grado B y cardinalidad final= cardinalidad A*cardinalidad B
- **Combinación**: Subconjunto del producto cartesiano que cumple una cierta condición que relaciona un atributo de la relación A con otro de la relación B. grado final=grado A+grado B y cardinalidad final<= cardinalidad A*cardinalidad B
- **Combinación Natural**: Consiste en una combinación en la que la condición es la igualdad de dos atributos, de distinta relación. Solo se mantiene uno de los dos atributos en los resultados.

Los operadores derivados pueden ser sustituidos por una secuencia de otras operaciones, sin embargo los operadores primitivos no se pueden sustituir por una expresión equivalente.

- Primitivos: Selección, Proyección, Producto Cartesiano, Diferencia y Union.
- Derivados: Intersección, Combinación y Combinación Natural.

Operadores de Álgebra Relacional extendida:

- **Agrupación**: Se obtienen los diferentes valores del atributo de criterio. Se añadirán atributos con funciones de agregación que se añaden como proyecciones o selecciones, pueden ser:
  - Count() conteo, Sum() sumatorio, Avg() media, Min() mínimo, Max() máximo. Todas esperan parámetros.
- **División**: Da lugar a las columnas de A que están concatenadas con el/los atributo/s indicados de la relación B. Y las filas se quedan las que tienen en la columna común los mismos valores.
- **Semi-Combinación**: Combinación natural, con un atributo que ambas relaciones tienen en común, en la que la relación final solo se queda los atributos de una de las relaciones, aparecerán solo las filas que tienen algún valor también en la otra relación.
  - Izquierda |*
  - Derecha *|
- **Anti-Combinación**: Es la opuesta a la semi-combinación, nos quedamos con las filas que no tienen ningún valor en común con la columna que comparten ambas. Nos podemos quedar con los atributos de la relación de la derecha o con los de la izquierda.

- **Combinación externa**: Combinación natural en la que también aparecen las filas que no tienen pareja en la otra relación, como no aparecía en ella en la final los atributos que faltan toman valor null. Podemos quedarnos solo los atributos de la derecha, de la izquierda o con todos. Si nos quedamos con un lado aparecen todos las filas de ese lado añadidas los atributos de la segunda relación con su valor si lo tenía asociado al criterio, si no null.

- **Orden**: Devuelve una lista ordenada en orden ascendente (menos a más) o descendente según la definición de la operación y criterio, se pueden aplicar funciones de agregación:

    - first (el primer valor de la lista), last (el último valor de la lista) y rank(valor) el de la posición valor.

Operador de Álgebra Relacional completa: Asignar y modificar.

- **Asignación**: Define operaciones de actualización sobre la base de datos.

    - Borrar filas: Mediante la resta de una relación y una parte de ella.
    - Borrar relación: Asignar vacío a la relación.
    - Añadir una fila: Union de la relación y una tupla con unos valores definidos.
    - Insert masivo: Union de una relación de con otra de la que hemos elegido las columnas que hacen que ambas sean compatibles, tengan los mismos atributos.
    - Actualizar datos: Union de la parte modificada con la relación sin esa parte.

Dinámica del SQL: DML

- **Modos de manipulación:**

    - Interactivo, mediante operaciones directas en SQL.
    - SQL embebido, mediante un lenguaje anfitrión, como C o Java.
    - Módulos, llamadas a procedimientos desde procesos externos.

- **Operaciones de actualización:**

    - Inserción de tuplas: Insert
    - Borrado de tuplas: Delete
    - Modificación de tuplas: Update

- **Operaciones de recuperación:**

    - Consulta o Query: Select

Lenguaje de control: LCD

- **Transacción**: Operación atómica sobre la base de datos. Conjunto de instrucciones de actualización.
- **Son**:
  - Commit [work], realizar, perpetra los cambios y son permanentes, si no se hace se perderán los datos.
  - Rollback [work] [to [savepoint] <savepoint>], deshacer, copia los datos de la BD a mi segmento privado, por lo que se restaura lo original y se piérdele progreso.
  - Savepoint <savepoint>, para que el rollback solo borres hasta ese punto.

Sintaxis de la QUERY: Todo se escribe en la misma sentencia, y lo que está entre [] es opcional

- **[WITH** <símbolo> **AS** <subquery>, ...]
  - Preclausula o renombrado, para representar una subquery con un solo símbolo, para que quede más limpio y se entienda mejor. Se pueden encadenar, separando con , .
- **SELECT [ALL|DISTINCT]** <lista de selección>
  - ALL, todas pudiéndose repetir, o DISTINCT, los valores únicos. Obligatorio estar seguido de las columnas que queremos consultar, * para todo el área de trabajo, atributos, pseudo-columnas(ROWNUM fila que ocupa esa posición y table. ROWID dirección física de la columna), constantes o variables ligadas, funciones aritméticas(+,-), de strings(||, substr), de codificación(case, nvl), de conversión (to_char, ...), del sistema(sysdate, user), de agregación o compiladas(creadas por el user). Se puede hacer renombrado.
  - **FROM** <cláusula de origen>
    - Indica la tabla o combinación de ellas(mediante otro SELECT, X CROSS JOIN Y, X JOIN Y [ON condición], X JOIN Y USING columnas con el mismo nombre, UNIÓN o {LEFT|RIGHT|FULL} JOIN [USING ...| ON ...]) de la que obtener los datos. Si no se quiere ninguna tabla se pone DUAL, es una tabla fantasma.
  - **[WHERE** <condición>**]**
    - Selección, ponemos la condición y aparecerán las que la cumplan. Mirar diapos.

- **[GROUP BY** <expresión> **[HAVING** <condcn>**]]**
    - Ordena según expresión, y en having están las condiciones colectivas.
- **[{UNION|UNION ALL| MINUS| INTERSECT}** <query>**]**
    - Union sin repetición, union con repetición, resta, intersección y query con la que realizar la operación, para hacer operaciones de más términos hay que ir encadenando query's que elijan la misma operación.
- **[ORDER BY** <expresión> **[ASC|DESC]] ;**
    - Lista ordenada en base a la operación, por defecto lo hace ascendente. Se acaba la sentencia con un ; .

4. TEMA 4: RELACIONAL AVANZADO

Desde el standard query language hasta el SQL3 se ha incorporado:

- Extensiones procedimentales: Se han añadido a la base nuevas funcionalidades.
- Comportamiento activo: Capturar eventos de datos, para que cuando ocurra cierto evento sé de una determinada acción. Trigger
- Diseño externo: Control de privilegios, vistas, ... Es la propia arquitectura de la base de datos: Conceptual, interno y externo.
- Diseño físico: Indices, clústeres...

Vista: Definición de tabla de lo que el usuario puede ver (no todos los usuarios ven todas las tablas, ni tampoco todas las filas), es una redundancia controlada(ya que es lo mismo que la tabla), en realidad la tabla solo estará una vez y se actualizara si se modifica en la vista.

- Debe ser operativa, que permita borrar, modificar e insertar, que se realizan sobre las tablas fuente.
- La BD se hace cargo de las operaciones triviales, pero cuando no lo son el programado debe definir que debe hacer. El grado será entre 1 y el número de columnas de la tabla original, y la cardinalidad al menos una fila.
- Una vista puede ser física, que se halla materializado, aunque ocupa espacio se accede más rápido que si se hace sobre otra tabla e ira actualizando la tabla de la que proviene.
- **Creación**:
  - **CREATE [MATERIALIZED] VIEW <nombre de tabla>**
    - Materialized si es física, y el nombre es el de la vista.
  - **[(<nombre de columna> [, <nombre de columna>]...)**
    - Podemos cambiar el nombre de las columnas de la expresión de consulta.
  - **AS <expresión de consulta> [WITH CHECK OPTION]**
    - Tabla de donde se sacan los datos, indicando columnas y condiciones.
    - El check option es para controlar que puedan o no salir filas de la vista, si no pueden salir no permitirá realizar esa modificación.

Clases de relación:

- **Persistentes**: Solo se borran con una acción del usuario.
  - **Relaciones base:** Tabla con todos los datos. Es a nivel lógico.
  - **Vistas**: Redefinición lógica de una tabla sin datos propios. Es un esquema externo.
  - **Vistas materializadas**: Redefinición lógica de una tabla con datos propios. La redundancia está controlada y los datos actualizados. Es a nivel interno.
  - **Instantáneas**: Copia de los datos de una tabla en un determinado momento, tabla congelada, para gestionar los datos de ese instante y cuando se acaba la eliminamos.
- **Temporales**: Desaparecen al ocurrir determinado evento, como una transacción o cierre de sesión. Se borra cuando se cierra la sesión.

Gestión de privilegios:

- **Elementos**:
  - **Usuarios**:
    - **CREATE USER <username> IDENTIFIED BY <password>**
    - **[DEFAULT TABLESPACE <tablespace>]**
    - **[QUOTA <size> ON <tablespace>]**
    - **[PROFILE <profile name>]**
    - **[PASSWORD EXPIRE]**
    - **[ACCOUNT {LOCK | UNLOCK}];**
  - **Perfiles:**
    - **CRÉATE PROFILE <profile name> LIMIT <resources>;**
  - **Roles:**
    - **CRÉATE ROLE <rolename>**
    - **{NOT IDENTIFIED | IDENTIFIED BY <password>};**
- **Privilegios:**
  - **Conceder:**
    - **GRANT {<rolename> | <sys_privileges | ALL PRIVILEGES}**
    - **TO <users/roles> [WITH ADMIN OPTION];**
    - **GRANT { <object_privileges | ALL PRIVILEGES}**

        - **[(column [, ...])] ON [schema.] <object>TO <users/roles>**
        - **[WITH HIERARCHY OPTION][WITH GRANT OPTION];**
    - **Revocar:**
        - **REVOKE <privileges> [ON <object>] FROM <users/roles>;**

- **Estructura de un bloque:** Declaraciones, Cuerpo y Excepciones.
    - **[DECLARE**
    - **varname type; [...]]**
    - **BEGIN**
    - **<código procedimental>**
    - **[EXCEPTION**
    - **WHEN ... THEN ...; [...]]**
    - **END;**
- **Procedimientos**: Omite el DECLARE
    - **CRÉATE OR REPLACE PROCEDURE name(params) IS**
    - **Bloque de código;**
- **Funciones**: Omite el DECLARE
    - **CRÉATE OR REPLACE FUNCTION name(params) RETURN CHAR IS**
    - **Bloque de código;**
- **Invocaciones a procedimientos**: Se deben hacer dentro de un bloque o con la instrucción EXEC.
    - **BEGIN my_proc(“”); END;**
    - **EXEC my_proc(“”);**
- **Paquete:**
- **Disparador:** Son procedimientos que se ejecutan ante un determinado evento, definidas por el usuario. Hay dos maneras esta es la sencilla, la compuesta en la 17 del tema 4.
    - **CRÉATE OR REPLACE TRIGGER [<nombre>]**
    - **<tiempo activación acción>**

        - Cuando se lleva a cabo: AFTER, BEFORE o INSTEAD OF. El último en vistas.
    - **<evento disparador>**
        - Porque se dispara: INSERT, DELETE o UPDATE [OF <columnas>]
    - **ON <nombre tabla>**
        - Tabla en la que se aplica.
    - **<nivel de activación>**
        - Si se hace para cada línea o para la operación. FOR EACH ROW o STATEMENT.
            - Se usa :old y :new para referirnos a la fila antes o después de la operación.
    - **<bloque definiendo la acción disparada>**

- **Error tabla mutante:** Cuando una tabla no está estable, durante una operación, y se opera sobre ella. Para observarlo hay que eliminar o insertar varias filas. Ocurre con FOR EACH ROW, pero no con FOR EACH STATEMENT.
    - **Soluciones**:
        - Almacenar en una tabla temporal las operaciones que se van a tener que realizar, para que cuando termine y este estable se realicen, con un disparador de fila.
        - Con disparadores complejos.

- **Desactivar constraints y triggers**: Hay más maneras.
    - **ALTER TABLE <table-name> {DISABLE | ENABLE} CONSTRAINT <c_name>;**
    - **ALTER TRIGGER <table-name> {DISABLE | ENABLE} ALL TRIGGER;**

- **Disparadores DDL:** Sobre las tablas de metadatos.

- **Disparadores DB**: Para crear tablas de auditorias, para almacenar eventos.

5. TEMA 5. FICHEROS: INTRODUCCIÓN Y CONCEPTOS BÁSICOS

Punto de partida:

- **Enfoque lógico, eficacia**: El usuario ve archivos, que son colecciones de registros, que son agregaciones de datos.
- **Enfoque físico, eficiencia**: La máquina acceder a ficheros, que son secuencias de bloques (unidad de acceso al soporte), que son conjuntos de bytes.

Estructura Física vs. Lógica:

- La unidad es el registro, y la unidad subatómica es el campo.
- **Correspondencia físico-lógica a nivel de registro**: Consideración de tamaños.
  - **Registró expandido:** Registro lógico que abarca varios bloques(registros físicos).
  - **Bloque**: Cuando en un registro físico, bloque, caben varios registros lógicos.
  - **Factor de Bloqueo:** Numero de registros lógicos que caben en un bloque. Solo se da en organización consecutiva, en no consecutiva se usa el tamaño de cubo.
- **Correspondencia físico-lógica entre registros:** La organización de registros.
  - **Organización Consecutiva:** Los registros lógicos están uno tras otro, no espera al siguiente bloque sino cabe entero. Se dice de tamaño n bloques.
  - **Organización No consecutiva:** Cuando un registro lógico, si no cabe completo en el bloque, se pasa al siguiente bloque. Se dirá fichero de n cubos.
- **Cubo:** Conjunto de bytes con una condición de acceso común. Para la no consecutiva es un conjunto de bloques que son utilizados como almacenamiento. Si se usan cubos pasaran a ser la unidad mínima, y se empezara a leer y escribir cubos.
  - **Espacio de cubo:** Información asignada a cubo.
  - **Tamaño de cubo:** Registros lógicos que caben en un cubo. Se redondea a la baja.

- **Partes de un cubo:**
  - **Información de control:** Cabecera, directorio de cubo, puntero encadenamiento, ...
  - **Espacio reservado(libre distribución):** Se reserva por si hay que modificar algún registro y aumenta el tamaño, así cabe y no hay que reescribirlo en un nuevo cubo
  - **Espacio para datos**: Hay espacio ocupado y espacio libre para añadir datos de los registros que hay dentro.

- **Diseño de Ficheros:**
  - **Diseño Lógico:** Descripción y disposición de los elementos de datos de un registro, que en conjunto definen un individuo.
    - **Unidad subatómica**: El campo, unidad mínima e indivisible.
      - Notación: camp tipo(tamaño)
    - **Agregado de datos:** Colección de elementos. Elemento de datos de evaluación múltiple.
      - **Vector**: Numero fijo de elementos que definen un concepto. Ejem: Fecha
        Notación: (elemento1; elemento2; elemento3; ...)
      - **Grupo repetitivo**: Compuesto por un numero fijo o variable de elementos cuya interpretación es común. Ej. Los hijos, puede haber 1 o x, pero con la misma estructura.
        Notación: $(elemento)^*$ desde 0 hasta N elementos.
      - Notación: $(elemento)^+$ desde 1 hasta N elementos.

- **Diseño Físico-lógico (Físico del registro lógico):** La implementación de un registro lógico en secuencia de bytes que permiten su lectura y escritura. Descripción de las cadenas de bytes utilizadas para almacenar registros.
  - Se busca la eficiencia, reducir el espacio por lo tanto el numero de accesos.
  - **Volumen y ocupación de un registro:**
    - **Volumen**: Numero de caracteres necesarios para almacenarlo.
    - **Ocupación útil:** Caracteres útiles del registro, los que usa de lo que le dan.
    - **Densidad ideal de un registro**: Relación entre cantidad de información útil y la cantidad de información almacenada. d= útil/real

- **Optimización**:
  - **Campos de control**: Marcas, mejoran el manejo que permiten ahorrar el relleno (padding) cuando ocupa menos de lo que se le da. La marca es un carácter, por lo que es un byte, 8 bits, que depende lo que almacene puede o no merecer la pena su uso.
    - **Elementos de datos:**
      Existencia: Indica en un campo opcional, si esta o no.
      Longitud: Índica la longitud en numero de caracteres.
      Reiteración: Numero de ocurrencias en un grupo repetitivo.
      Fin de Campo: Indica cuando acaba un campo, se usa para campos muy grandes. Su uso es peligroso.
    - **Registro**:
      Fin(inicio) de registro: Separa registros consecutivos.
      Tipo: Indica el tipo de registro a continuación.
      Mapa: Indica los registros que se aplica, agrupación de bytes.
  - **Codificación de campos:** Consiste en sustituir una información por otra equivalente de menor tamaño. Algunos:
    - Utilizar codificación numérica.
    - Utilizar enumerados.
    - Fecha en formato Juliano (la de un numero).
    - Agrupación de varios campos.

- **Diseño Físico:** Disposición física de los registros en el soporte, para acceder a ellos con el menor coste y números de accesos que sea posible. El número de accesos es el más importante, porque estos conllevan también tiempo de acceso.

  - **Espacio de un fichero**: Se busca que ocupe lo mínimo. La densidad ideal es menor o igual que la real casi siempre, tenerlo en cuenta al obtener resultados.
    - **Densidad real(dr) de un fichero**: Relación entre cantidad de información útil y cantidad de información almacenada. Esta medida es más global que la ideal, ya que la ideal se limita a un registro. La **fórmula** desglosada es: el número de registros por lo que usamos de los registros para almacenar datos, sin información de control y libre, partido del numero de bloques por el tamaño de los bloques.
    - **Densidad de ocupación(do) de un fichero no consecutivo**: Relación entre cantidad de registros almacenados y cantidad a de ellos que caben, el espacio potencialmente útil. **La fórmula** es el número de registros que hay partido del numero de cubos por el número de registros por cubo.

- **Coste Global:** Es la media ponderada de accesos lógicos(por unidad de tiempo o carga) de todos los procesos. Hay que hallar el coste para cada tipo de organización y observar cuál es la que mejor se adapta. Una organización física del Sistema de Archivos (O) define todas las organizaciones base de los archivos que incluye. Cada proceso P, tendrá en O un coste C asociado, que se expresa en numero de accesos o tiempo.
    - Todos los sistemas de archivos están sometidos a un conjunto de procesos, y estos procesos tienen una frecuencia asociada referida a una unidad de tiempo(segundos, horas, ...) La suma de todos los procesos de un sistema es 1.
- **Coste a bajo nivel:**
    - Determinados soportes mejoran el acceso serial.
    - Pueden almacenar bloques en memoria privilegiada (en páginas)
        - En memoria intermedia (buffer), es que es más rápida(ahorra accesos), pero más costosa(es escasa)
    - **Hit ratio(hr|ϕ):** Porcentaje de accesos ahorrados por la memoria intermedia. Las veces que pide y está en el buffer, y no tiene que traer.
        - **coste real(efectivo)= (1-hr)*coste global**
    - **PIO(Physical Input Output):** Numero de veces que físicamente se lee el bloque. Es el que se prioriza reducir, pero es más difícil, por lo que se hace por medio del LIO. LIO y PIO son proporcionales.
    - **LIO(Logical Input Output)**: Numero de veces que pide una página, este o no en buffer. LIO y PIO se relacionan mediante el hit ratio.
- **Interacción con FF(Ficheros)**
    - **Clave**: Campo o secuencia de campos, en un orden, con una función especifica en la interacción de los usuarios con el fichero. Tipos:
        - **De identificación:** Campo o conjunto de campos que identifican unívocamente un registro. Sin valores repetidos.
        - **No identificativa**: Lo contrario. Presenta valores repetidos en el fichero.
          Coincidencia k: Tasa media de registros que toman un determinado valor. $k = \frac{r}{\#valores}$. r numero de registros del fichero, que lo toman.
          Cardinalidad del dominio: Numero de valores distintos. #valores.
          De búsqueda: Campo o conjunto de campos que es frecuente para realizar una búsqueda. No solo son las del where sino también las de proyección.
          Privilegiada: Campo sobre la que existe un mecanismo físico que hace la recuperación más eficiente.
          De direccionamiento: Determina la ubicación del registro.

De ordenación: Criterio físico o lógico de ordenación.
De indización: Privilegiada mediante una estructura auxiliar.
De agrupación: Reúne registros con esa clave común. No Group By

- **Tipología de Procesos:**
  - **Diferenciar entre:**
    - **Actualización:** Implican escritura.
    - **Recuperación**: Solo implican lectura.
  - **Diferenciar entre:**
    - **Selectivo:** Aquel que impone un filtro o condición, solo a esos.
    - **Incondicional**: No impone condiciones, se refiere a al totalidad.
  - **Diferenciar entre:**
    - **Identificativa(exact match)**: Aquella cuya clave de búsqueda es identificativa, que encuentra a solo 1. De media recorre la mitad.
      Simple: Si es la clave identificativa.
      Multiclave: No tiene sentido, la identificativa ya lo encuentra.
    - **No identificativa:** Hay que recórrelos todos, ya que hay más de uno, pero no sabes o cuantos.
      Simple: k registros (no identificativa). k registros en un rango.
      Multiclave: k registros en un rango (window query). Proyección de pocos atributos del resultado de una WQ.
- **Conjunto de Direcciones Relevantes**:
  - Aquellas claves privilegiadas que permiten filtrar, y tras descartar las que no lo cumplan se recorren las restantes, a menos que se encuentre lo que se busca (de media (N+1)/2).
  - Se pueden utilizar booleanos para indicar que registros debemos recorrer.
  - Una de las estrategias de filtrados es, hacer un árbol de decisiones y la rama escogida es el plan de ejecución.

6. TEMA 6. ORGANIZACIÓN DE FICHEROS: ORGANIZACIÓN BASE (FÍSICA)

Introducción:

- **Que buscamos optimizar:**
  - **Tiempo de respuesta:** Disminuyendo el número de accesos y evitando reorganizaciones en tiempo de proceso, etc.
  - **Espacio de Almacenamiento**: Incrementar la densidad, minimizar almacenamiento de estructuras auxiliares, etc. Este también afecta al tiempo de respuesta.
  - **Coste de desarrollo y mantenimiento**.
- **Partimos de ciertos requisitos:**
  - **Características del dispositivo**: Bloque(tamaño), tiempo acceso(secuencialidad), etc. Son las características físicas.
  - **Características de los archivos**: Tamaño registro, cardinalidad, volatilidad, etc.
  - **Características de los procesos:** Tipología, frecuencia, criticidad (aquellos que tengo en cuenta para optimizar, los que marcan la diferencia)

Organizaciones consecutivas:

- **Organización básica:** Organización serial.
  - Registro físicos en serie, están uno detrás del otro y se acceden en ese orden. La inserción se hace siempre al final, sin ningún criterio de colocación. Consecutivos: n bloques. No consecutivos: N cubos.
  - El último cubo es privilegiados al ser el más frecuente, ya que es el que se accede más, y lo mantengo en memoria intermedia, de esta manera es más rápido el acceso y nos ahorra tener que buscar el último a la hora de insertar.
  - **Características**:
    - Aprovechamiento de espacio y Coste de accesos a la totalidad, ÓPTIMO. Ya que están amontonados, por lo que están cercanos.
    - No existen claves privilegiadas, no se puede filtrar.
    - Tamaño área de búsqueda: n bloques (consecutivos) y N cubos (no-consecutivo)

- **Procesos de organización serial:**
  - **Actualización**: Implica escritura.
    - **Inserción**: Se añaden los registros al final del fichero.
      Coste 1 acceso normalmente, si es un registro reciclado implica 2 accesos.
    - **Borrado**: Eliminar un registro. Coste para todo tipo: selección + k accesos
      Borrado físico: Se da en la organización no consecutiva, se borra el registro y se recolocan el resto de registros del cubo.
      Borrado lógico: Se da en la organización consecutiva, se hace una marca en el hueco del registro que se quiere eliminar indicando que tamaño tiene para que en la lectura se pueda saltar. En realidad no sé eliminar el contenido, solo se hace la marca. Este borrado no desplaza los registros tras el borrado, sería muy costoso.
    - **Modificación**: Casi siempre serán para ampliar el tamaño.
      Registros fijos: Es la que se produce en registros que tienen siempre el mismo tamaño, por lo tanto se hace en ese hueco la modificación.
      Serial no consecutiva: Dependerá de si cabe o no, si hay ELD
      Si hay hueco, se modifica en el mismo cubo, el ELD lo permite.
      Si no hay hueco, se busca otro cubo en el que quepa, muy probablemente sea el último.
      Coste en registros fijo o en no consecutivos: selección + k accesos.
      Otros casos: Se borra el antiguo y se reinserta modificado.
      $Coste : Seleccin + k + k$
  - **Recuperación**: Es la lectura de registros.
    - **Consulta selectiva identificativa:** Leemos hasta encontrarlo, solo hay 1, por lo que una vez encontrado termina.
      $Coste : (N + 1)/2$ en no consecutiva y $(n + 1)/2$ en consecutiva.
      N numero de cubos y n numero de bloques.
    - **Consulta selectiva no identificativa:** Leemos todos los registros, ya que no sabemos cuantos habrá que cumplan esa clave.
    - **Consultar selectiva multiclave:** Leemos todos los registros, ya que no sabemos cuantos habrá que cumplan esa clave.
    - **Consulta a la totalidad:** Leemos todos los registros, ya que no sabemos cuantos habrá que cumplan esa clave.
    - **Coste en el resto:** N en no consecutiva y n en consecutiva.
- **Mantenimiento Serial:** Gestión de huecos.
  - **Espacio Libre Distribuido:** Se mantiene un espacio libre, para que si hay que realizar una modificación esta quepa en el cubo, y no halla que buscar otro. Ya que es muy costosa la modificación si cambia de ubicación, hay que actualizar todo.

        - ◇ Mover registros es engorroso y genera costes adicionales.
        - ◇ En organizaciones consecutivas, se puede evitar dejando espacio libre distribuido, que es un porcentaje de espacio del cubo reservado para modificaciones.
        - ◇ En Oracle se reserva un 10 %.
    - **Gestión de Huecos o Pila de inserción:** Lista de cubos candidatos para insertar, los mantiene localizados. Son aquellos que tienen la ocupación por debajo del umbral de ocupación, que nos indica que el registro puede ser reciclado. Se reduce el tamaño del fichero, aunque pasara la inserción a costar dos accesos. Esta estructura se realiza en el primer full scan.
    - **Compactación:** Proceso que desplaza registros para eliminar grandes huecos producidos por el borrado o modificación. Refresca los cubos: Si falta hueco a ELD, se añade espacios y si falta se le quita. Se hace forma periódica en las organizaciones consecutivas y en las no consecutivas se hacen en el momento.

- **Organización ordenada:** Organización secuencial.
  - Surge de la organización serial introduciendo un orden.
  - El acceso aleatorio a bloques obliga a contar con un mecanismo para interpretar su contenido, localizar el comienzo del primer registro.
    - A nivel físico, cubos, comienza con un registro completo.
    - A nivel físico-lógico, registros consecutivos, comienza en una marca de inicio/fin.
  - Almacena registros con un criterio de orden.
  - **Características:**
    - El aprovechamiento de espacio y el coste de accesos a la totalidad son ÓPTIMOS.
    - En esta organización si hay una clave privilegiada de orden, que nos permita hacer búsqueda dicotomía.
    - Tamaño área de búsqueda: n bloques (consecutivos) y N cubos (no-consecutivo).
  - **Procesos de organización secuencial:**
    - **Actualización:**
      - ◇ **Inserción no ordenada:** Se añaden registros al final del fichero, lo que provoca que la organización degenere y se reduzca la eficiencia. Coste: 1 acceso, se introduce al final.
      - ◇ **Inserción ordenada:** Se localiza la ubicación del registro, para introducirlo, si no cabe provoca desbordamiento que requiere gestión de desbordamientos. Este tipo de inserción es más costoso, pero los mantiene ordenados.

$Coste : log_2(x+1)+1$

- ◇ **Borrado:** Se mantienen las dos posibilidades, hacer una marca en los consecutivos, y en los no consecutivos eliminar y colocar el resto de registros.
  $Coste : seleccin + k/T_c\ accesos$. Tc tamaño del cubo. Es encontrarlo y después todos los accesos a cubos para los k registros.
- ◇ **Modificación:** Se mantienen los de serial, los no consecutivos lo modifican en el mismo cubo y los consecutivos se borra e introduce ya modificado. La diferencia es que modificar la clave de ordenación hace que tengamos que eliminar el registro y reinsertarlo, para que ocupe la posición correcta.
  Coste en no consecutivas y no modifica Co: selección + k/Tc accesos.
  Coste otros casos: coste borrado + coste inserción

- ○ **Recuperación:** El coste de la sección (selección accesos)
- ○ **Selección:**
  - ◇ **Consulta por clave no privilegiada:** Recorre todos.
  - ◇ **Consulta por clave privilegiada**
    Clave identificativa: Mediante búsqueda dicotomía. Coste: log2(x+1)
    Clave no identificativa: Mediante búsqueda dicotomía extendida. Coste: coste de la búsqueda dicotomía extendida + coste desbordamiento.
  - ◇ **Consulta selectiva multiclave:** Primero filtra, luego hace búsqueda dicotómica.
    Coste: Primer filtra y luego búsqueda dicotomía.
  - ◇ **Consulta a la totalidad ordenada(por clave privilegiada):** Óptima.
    Coste: Coste serial de un full scan de N.

- **Búsqueda dicotómica**: Mira el elemento central del espacio de búsqueda, si coincide termina, pero si no restringe el espacio de búsqueda a la mitad. Y se repite el proceso.
- **Búsqueda dicotómica extendida:** Primero hace búsqueda dicotómica hasta encontrar uno de los elementos, después busca serialmente hacia delante y hacia atrás, hasta que encuentre un fallo por ambos lados.
- **Mantenimiento Secuencial:**
  - ○ **Gestión de desbordamiento:**
    - ◇ **En organización consecutiva:** Hay un área de desbordamiento, donde están desordenadas.
    - ◇ **En organización no consecutiva**:
      Rotación: Traspasa elementos de un cubo lleno a su vecino, si tiene espacio libre.

Intercalar cubos completamente vacíos, son virtuales, realmente no están en medio físicamente, pero actúan como tal.
Partición celular: Al desbordarse, se añade en medio un cubo vacío (también virtual) para que entre el desbordamiento y se mantenga en orden. Y se reparten los registros, entre el nuevo cubo y el desbordado.

- **Espacio Libre Distribuido para inserción:** Se usa en organizaciones secuenciales, para reducir la tasa de desbordamiento. Reserva una parte del cubo para posibles futuras inserciones o modificaciones.
- **Lista de huecos:** Se usa poco esta estructura que localiza los huecos, ya que estos tienen un orden, tamaño...
- **Reorganización o Reordenación:** Compacta y reescribe todos los registros ordenados, mezcla las áreas. Es una operación costosa, pero reduce el tamaño y mejora la eficiencia.

- **Organizaciones Direccionadas (Siempre es no-consecutivo)**

  - **El acceso aleatorio**: proporciona el registro físico indicado. Conociendo la clave de búsqueda puede conocer directamente su posición.
  - Clave privilegiada, la CD (Clave de Direccionamiento), tiene gran capacidad de filtrado máxima.
  - **Aprovechamiento de espacio:** reducido.
  - **Coste de accesos a la totalidad:** elevado.
  - **Espacio de direccionamiento:** N cubos.
  - **Tipos de Direccionamiento:**
    - **Organización Direccionada Directa:**
      - Cada registro tiene su dirección reservada y cada cubo un solo registro.
        Si el tamaño máximo del registro es mucho menor que el del bloque, se pueden considerar **celdas**.
      - Los valores de CD que no ocurren implican cubos vacíos. (baja densidad).
      - Se puede transformar la dirección, si solo ocurre en un rango de valores:
        Org. Direccionada Directa Absoluta: la CD es la dirección del cubo.
        Org. Direccionada Directa Relativa: existe una biyección entre CD y la dirección del cubo. Ejem: Truncar la clave.

- **Organización Direccionada Dispersa (HASH):**
  - La CD se transforma en dirección del cubo, pero la función no es una biyección. Si la distribución es buena, aumenta la densidad.
  - **Algoritmo de Transformación:**
    Conversión numérica: Se pasa la clave a un valor numérico. Ejem: ASCII
    Función de dispersión: Se encarga de repartir los distintos cubos en las posibles direcciones, de 0 a N-1. Busca que estén distribuidos uniformemente, sin que haya cubos vacíos o muchos registros en el mismo cubo.
  - Si la densidad es baja, se puede reorganizar cambiando:
    La dispersión, el diseño del cubo o el espacio de Direccionamiento.
- **Conceptos:**
  - **Claves sinónimas:** Producen la misma dirección.
  - **Claves homónimas:** Tienen el mismo valor.
  - **Potencia de Direccionamiento:** #valores(CD) >= N
  - **Colisión:** Inserción en un cubo no vacío.
  - **Desbordamiento**: Colisión en un cubo sin espacio suficiente.

- **Procs. Direccionamiento:**
  - **Actualización:**
    - **Inserción:** Calcula la dirección, y se añade el registro allí. Si no cabe, se desborda. Coste: 2 accesos.
    - **Borrado:** Localiza el registro y se elimina. Coste: selección + k accesos
    - **Modificación**: Localiza el registro y se modifica, si no cabe borra e inserta.
      No actualiza la CD: selección + k accesos.
      Actualiza CD: Borrado y reinserción.
  - **Recuperación:** El coste de la selección.
    - **Localización:** Filtra por CD
      Consulta selectiva identificativa: leer cubos no filtrados, hasta encaje.
      1 + Prob.desb.*(Ndesb.+1)/2 accesos.
    - **Consulta selectiva no identificativa:** leer todos los cubos no filtrados.
      1 + Ndesb. Accesos.
    - **Consulta selectiva multiclave:** Filtrado multiclave.
      $2^b + N_{desb}$. Accesos. b, numero de bits de la dir. que desconocemos.
    - **Otras consultas**: $fullscan = N + N_{desb}$.

- **Gestión Desbordamiento:**
  - **Dos criterios**
    - **Donde ser ubique el registro desbordado:**
      Saturación: Dentro del espacio de Direccionamiento.
      Área de Desbordamiento: Fuera del área de datos.
    - **Mecanismo de ubicación:**
      Direccionamiento abierto: la dirección nueva es la anterior más k.
      Encadenamiento: Se marca donde está el registro desbordado.
      Otros (organización independiente)
  - **Saturación con Direccionamiento Abierto:** La nueva dirección se averigua a partir de la dirección desbordada.
    - **Sondeo lineal:** $dir' = dir + 1$
    - **Rehashing**: $dir' = dir + k$
    - **Doble hash**: $dir' = f(CD)$
    - Si esta estuviera ocupada se produce un **choque**. Si además no cabe, será un **rebote**.
    - Si se produce un rebote ser buscara otra dirección nueva hasta encontrar una posición libre o hasta haber recorrido todo el espacio.
    - Se marca con un **Byte de desbordamiento** en el cubo que indica si ha desbordado, que nos permite saber cuando dejar de buscar.
    - En media ser recorren: N/(#cubosConByteDesbordamiento0 +1)
  - **Saturación Progresiva Encadenada:** La nueva dirección se deja apuntada en el cubo desbordado.
    - **Puntero**: Indica la ubicación de otra información. Puede ser lógico(clave), relativo(dir.) o físico.
      Puntero relativo de precisión simple: Indica en que cubo.
      Puntero relativo de precisión doble: En que cubo y en que posición. Es el que usa para este desbordamiento, al ser un registro específico.
    - Los registros que desbordan en un mismo cubo se van encadenando, el cubo apunta al segundo y el segundo al primero.
  - **Área de Desbordamiento Independiente:** Los registros desbordados son almacenados fuera del área de datos, en un archivo aparte.
    - **Ventajas**: Se eliminan los choques y rebotes.
    - **Desventajas**: Necesita más espacio, un área auxiliar. La organización degenera y baja la eficiencia. Este área no se filtra en búsqueda por clave privilegiada. Y si es por clave alternativa, se tienen más cubos.
    - En el área de desbordamiento normalmente la organización es serial, y si desborda el área de desbordamiento se hace otra gestión.

- **Encadenamiento en Área de Desbordamiento**: Los registros que desbordan se almacenan en un área aparte, y su dirección se deja apuntada en el cubo desbordado.
    - **Encadenamiento a registro**: Los registros en área de desbordamiento se almacenan serialmente, pero incorporan un puntero de encadenamiento.
    - **Encadenamiento a cubo (Extensión del cubo de datos)**: Cuando un cubo desborda, se le asigna a esa dirección un cubo completo dentro del área de desbordamiento:
      El puntero de encadenamiento es de precisión simple.
      El cubo solo contiene registros de la dirección que lo apuntan.

- **Dir. Disperso Multi-Clave**: Consiste en ampliar el algoritmo de transformación, para operar varias CD. Combina las dispersiones de varias CD, cada una con su espacio de direccionamiento y función de dispersión.
    - Aunque no conozcamos todas las CD, podemos filtrar solo con conocer uno de esos atributos. El algoritmo de filtrado utiliza bucles anidados.
- N son la cantidad de direcciones que representa cada CD.
- Cuantos más bits se le asigne más filtra la CD.
- **Mantenimiento Hash**:
    - El encadenamiento a cubo proporciona extensiones automáticas de espacio para cada dirección con poco coste.
    - Por tanto, definir ELD para inserción no es (en general) una ventaja, pero si un N más grande.
    - Si el área de datos está demasiado saturada o demasiado vacío, es necesario reorganizar. Aunque la reorganización automática no suele ofrecer buen rendimiento.
- **Clúster de datos:** Agrupación física de registros que tengan el mismo valor para una clave privilegiada (Clave de agrupación).
    - Puede haber registros de distintos tipos, pero con el atributo común.
    - Son una **organización no consecutiva** y puede ser:
        - Simple: Serial.
        - Indizado: Serial con índice sobre la clave de agrupación.
        - Disperso: Mejora procesos selectivos, por CD.
        - Ordenado: Mejora procesos selectivos y ordenados.
    - CREATE CLUSTER nombreCluster (nombre TIPO(Tamaño));
    - CRÉATE TABLE nombreTabla... CLUSTER nombreCluster(AtribTabla);
    - Mirar diapositivas para las características en Oracle.

7. TEMA 7: ORGANIZACION DE FICHERO: ORGANIZACIONES AUXILIARES

Índice: Archivo aparte donde se almacena la ubicación física de los valores de una clave alternativa que es muy frecuente. Es un directorio cuya entrada se refiere a un solo registro.

- El archivo auxiliar es el **Índice** y la clave privilegiada es la **Clave de indización**.
- **Tipos de punteros**: De menos a más independiza y de más a menos velocidad.
  - **Dirección de Máquina:** la dirección física del registro.
  - **Dirección Relativa:** Registro en el espacio de Direccionamiento.
  - **Puntero simbólico**: Identificación lógica del registro.
- **Entrada**: Registro formado por punteros.
  - Entrada de **índice primario**: $clave * puntero\_externo$
  - Entrada de **índice secundario**: $clave * long\_lista * (puntero\_externo)^{long\_lista}$.
  - El número de entradas es igual a la cardinalidad del dominio(CI).
  - Al buscar recorremos el índice hasta encontrar una entrada, y si insertamos se pone al final de la lista de punteros. Conviene que el índice esté ordenado, y no consecutivo.
- **Directorio**: Archivo formado por entradas.
- **Tipos de índices:**
  - **Primario**: La clave de indización es identificativa. 1 entrada - 1 clave - 1 registro. Filtrado máximo.
  - **Secundario**: La clave de indización es no identificativa. N registros - 1 clave - 1 entrada. Filtra menos.
- **Ventajas**:
  - Acceso por clave alternativa, hasta ahora no privilegiadas.
  - Aumenta la Tasa de Acierto, al ocupar menos y ser recurrente la mantenemos en memoria.
  - Reorganización menos costosa, los índices tiene menos bloques que los propios datos.

- **Desventajas**:
  - Procesos de Actualización más costosos. (Muy importante)
  - Necesita almacenamiento auxiliar.
  - Necesita mantenimiento.
- **Operaciones**: Creación, Borrado, Consulta, Localización y Actualización.
- **Coste de Procesos sobre Ficheros Indizados:**
  - **Localización a través del Índice**: Acceso al índice.
  - **Localización por varios índices:** Suma del acceso a cada índice.
  - **Recuperación**: $acceso\_indice + acceso\_datos$.
  - **Actualización**:
    - **Inserciones**: Inserción de entradas. **Coste = acc_indice + 1.**
    - **Borrados**: $Coste = acc\_indice + 1$
      - **Indice primario:** suelen requerir borrado de entradas, se vacía al ser 1 solo.
      - **Indice secundario**: pueden requerir modificación de entradas, si se vacía.
    - **Modificaciones**:
      - **CI**: Suele implicar borrado + reinserción de entrada. $2 * acc\_ind + 2$
      - **CD/CO**: Cambia ubicación reg., cambia puntero. $acc\_indice + 1$
- **Taxonomía de índices:**
  - Según el **carácter de la clave de indización**:
    - **Índices primarios** vs. **índices secundarios**.
  - Según la **correspondencia entre entrada y registros**:
    - **Denso**: Una entrada del índice para cada registro.
    - **No denso**: Una entrada para cada cubo de datos.
  - Según el **recubrimiento del índice**:
    - **Exhaustivo**: Todos los registros tienen una entrada.
    - **Parcial**: No se indizan todos los registros.
  - Según la **estructura**: Índices simples vs. indices multinivel.

- **Indice Simple Denso:**
  - **Naturaleza**: serial, secuencial, o Direccionada.
  - **Coste**: Depende de la naturaleza.
  - **Restricciones**: Sobre claves no privilegiadas.
  - **Mantenimiento**: Ordenado o disperso, puede desbordar —> Reorganización.
    - Se debe evitar la degeneración de la estructura:
      - **Indice ordenado**: Preferible inserción ordenada + reorganización local.
      - **Indice disperso**: Pierde eficiencia si cambia, es más útil como indice temporal.
      -
- **Indice Simple No Denso**: Una entrada por cubo de datos.
  - **Restricción**: indice y organización base deben ser necesariamente secuenciales y con clave_indizacion = clave_ordenacion (CO=CI)
  - **Usos**: Varias posibilidades de acceso:
    - **Procesos ordenados** (a la totalidad): Acceso serial.
    - **Procesos selectivos** (solución única): A través del índice.
    - **Mixtos**: (selección de un rango): acceso indizado + serial.
  - **Ventajas**:
    - Tamaño muy reducido, tiene menor coste y mayor tasa de acierto. (Apunta a cubo)
    - Se ahorra muchas actualizaciones de índice.
    - Se pueden utilizar prefijos, el mínimo tamaño para reconocerlo, en lugar de utilizar toda la clave en la entrada.
  - **Inconvenientes**:
    - **Solo puede existir un índice no denso para cada archivo**.
    - La inserción del registro debe ser ordenada, pero se localiza con el índice.
    - La inserción de la entrada es ordenada, y conlleva pesadas reorganizaciones.

- **Indice Multinivel**: Es un índice con n niveles, árbol de índices.
    - El coste es un acceso por nivel, interesa definir nodos pequeños a costa de tener más niveles. Bloquear la raíz en memoria intermedia, nos ahorra un acceso.
    - El último nivel, n, suele ser denso. Aunque al ser secuencial puede ser no denso.
    - Es eficiente, pero degenera.
    - Para fichero constante es buena solución, pero cuando son volátiles requiere reorganización.
    - **Posibles soluciones**:
        - **Árboles binarios:** Presenta problemas de vecindad y desequilibrio.
        - **Árboles AVL**: Resuelve el desequilibrio con reorganización local.
        - **Árboles Binarios Paginados**: Almacena en cada nodo sus dos hijos, 2 niveles.
        - **Árboles AVL- Paginados**: Buen rendimiento, pero necesita muchos punteros internos, bajísima densidad y reorganizaciones frecuentes.
        - **Árboles B**: La mejor solución. Incluye varias entradas por nodo y se construye en orden ascendente.

Indización en Árboles B: Comienza por las hojas y se va construyendo hacia arriba.

- **Nodo**: Contiene entradas de índice(CI-puntero) y punteros(a hijos).
- **Orden del árbol (m)**: Capacidad de los nodos, según los punteros, el número de hijos de un nodo.
- **Corolario**:
    -
    - **k = m - 1**. Si un nodo tiene m descendientes, tendrá m-1 entradas.
    - El nodo raíz tiene al menos un elemento y por lo tanto al menos dos hijos.
    - El tamaño de nodo es múltiplo del tamaño de bloque.
    - El **tamaño de la entrada**, es el de la **clave más el/los punteros internos**.
- **Partición y promoción**: Las entradas de un nodo están ordenadas, y cuando desborda, se divide en dos nodos y se promociona el elemento intermedio hacia el nivel superior.

- **Propiedades**:
  - **Recuperar una entrada**: #accesos = #niveles.
  - **Recuperar un registro aleatorio**, se recuperan la entrada y tantos cubos de datos como punteros tenga la entrada. $Coste = (n-1) * T_{nodo} + c * E_{cubo}$.
  - El **coste de cualquier actualización sobre el índice**, es el coste de localización más un acceso de escritura: $(n-1) + 1 = n$
  - El **coste extra de una partición** es de dos accesos de escritura.
- **Aspectos positivos**:
  - Es bueno para índices primarios, pero no tanto para los secundarios.
  - Las reestructuraciones son locales.
  - Las páginas están ocupadas a la mitad, hay espacio para cambios.
- **Aspectos a mejorar:**
  - La densidad de los nodos es baja.
  - Los punteros de las hojas no son necesarios.

Indización en Árboles B^*: Pretende aumentar la densidad.

- En lugar de dividir un nodo en dos, se dividen dos nodos en tres. Pasa del 50 % al 66 %.
- Cuando un nodo desborda, lo primero que se intenta es hacer una rotación a los nodos vecinos, pasando a ser el discriminante.
- Si también se llena el nodo vencido es cuando se pasa de dos llenos a tres nodos.
- Todo lo demás funciona como en los árboles B.
- **Ventajas**:
  - Aumento de la densidad.
  - Un desbordamiento, no siempre supone partición.
- **Desventajas**:
  - Aumenta la probabilidad de desbordamiento, al estar de media más llenos.

- **Propiedades**:
  - Todos los nodos menos el raíz garantizan una ocupación mínima: $k_{min} = \lfloor \frac{2k}{3} \rfloor$
  - Los nodos intermedios cuentan con $(2k/3) + 1$ descendientes: $m_{min} = \lfloor \frac{2m+1}{3} \rfloor$
- **Cálculo de costes:** Saberlo como teoría, pero no como práctica.
  - La localización es idéntica al árbol B.
  - Coste extra de hacer **Rotación**: 3 accesos.
  - Coste extra de la **Partición**: 4 accesos.

Indización en Árboles B^+: Coste proporcional a la profundidad, crecer en amplitud, aumenta el número de hijos por nodo, aumenta el orden.

- Los nodos con hijos suprimen los punteros externos(apuntan a registros), por lo que caben más punteros internos, es decir más hijos. Son solo punteros a otros nodos.
- En los nodos hoja no hay punteros a nodo hijo, pero sí habrá punteros externos, para apuntar a los datos.
- En las hojas se usa un puntero interno adicional para apuntar al siguiente nodo hoja. Es un mecanismo de acceso alternativo. El la partición se hace:
  - El puntero de encadenamiento nuevo apunta al nodo viejo, y el viejo apunta a al dirección del nodo nuevo.
- Especialmente eficiente con punteros externos grandes, indice secundario.
- La partición/promoción es igual que en nodos de árbol B.
- **Propiedades**:
  - **Orden del árbol (m):** Se calcula para nodos no hoja, como en árboles B, pero teniendo en cuenta que las entradas no contienen punteros externos.
  - **Ocupación máxima (k) de los nodos hoja:** Si los tamaños de los punteros internos y externos son distintos.
  - **Ocupación mínima de las hojas será**: $k_{min} = \lfloor \frac{k+1}{2} \rfloor$
  - **Ocupación mínima de los nodos intermedios será:** $m_{min} = \lfloor \frac{m+1}{2} \rfloor$
  - **Cálculo del numero de niveles:**
    - $numero\ de\ hojas = \lfloor \frac{e}{k_{min}} \rfloor$ tal que e=numero total de entradas.
    - numero de $nodos(n-1) = \lfloor \frac{nodos\ Nivel\ n}{m_{min}} \rfloor$
    - Cuando llega un nivel con un solo nodo, es la raíz nivel 1.

        - **Tamaño máximo del fichero indice:** Es la suma de los nodos necesarios para cada nivel multiplicado por el tamaño de un nodo.
    - Las mejores logradas con árboles B+ y B* son combinables.

Estructuras especiales:

- **Indice intermedio:** indice primario, denso y exhaustivo, cuyos punteros apuntan a los datos, y el resto de los índices apuntan a este. Al cambiar los registros de ubicación, solo es necesario actualizar punteros.
    - Es muy eficiente, está bloqueado en memoria privilegiada, es de tamaño reducido y casi constante.
- **Indice agrupado o Clúster:** Dos o más indices sobre distintos archivos con la misma CI y valores validados pueden combinarse. La entrada tendrá una clave de indización y uno o más esquemas de punteros.
- **Indice Multiclave: el árbol R.** Admite la creación de índices especiales que no estén basados en una clave. Un árbol R, es una evolución del árbol B+ para d dimensiones.
- **Esquemas de bits(BITMAP):** Para dominios con cardinalidad pequeña. Un esquema de bits para un campo es un vector de valores booleanos. A cada valor del dominio se le hace corresponder una posición.
    - Puede ser simple o multiclave, concatenando esquemas de varios campos.
- **Máscaras sobre BITMAP:**
    - **Máscaras para condiciones de igualdad:** Un bit para cada posible valor, 0 o 1, bivaluada. Filtra los que tienen exactamente un 1 o 0. S and Q = Q
    - **Máscaras con bits que admitan cualquier valor**: Un bit para posible valor, 0, 1 o q, trivaluada. Permite filtrar valores exactos y otros que no tener en cuenta, q.
        - S XOR - Q = 1
- **Esquema de bits Simple vs. Multiclave:** Es conveniente que el diseño de los esquemas de bits se realice atendiendo a las necesidades de procesamiento.

Acceso Invertido: Es un tipo de acceso indicado multiclave orientado a optimizar el coste de acceso en procesos muy concretos. Trata de averiguar información delimitada de ciertos archivos con condiciones muy concretas.

- Procura averiguar toda esta información accediendo solo a los índices, sin llegar a acceder al dato. Deben localizarse unívocamente cada registro.
    - Se ejecutarán primero las condiciones y después se busca en los índices objetivo, pero observando los indices en vez de las entradas y lo que buscamos es la entrada par un determinado indice.
- Es eficiente si requiere acceder a pocos índices. Los índices que soporta este acceso se denominan **índices inversos**. Los secundarios también se denominan **listas invertidas**.
- Un **fichero invertido** es el que soporta este tipo de acceso, y se llama totalmente invertido si todos los campos invertidos.
- **Costes**: n es el número de bloques del índice. r el número de resultados.
    - **Selección**:
        - **Listas invertidas no ordenada**s: El máximo es n, y el medio $\frac{(n+1)}{2}$.
        - **Listas invertidas ordenadas**: $\log_2(n+1)$
        - **Esquema de bits**: n.
        - **Otro tipo de índices**: Depende de la estructura.
    - **Proyección**:
        - **Esquema de bits con puntero implícito**: mín(n, r)
        - **Cualquier otro caso**: n.

8. TEMA 8: SISTEMAS GESTORES DE BBDD

Acceso en Drive a las diapositivas

9. TEMA 9: PARADIGMAS BD - BIG DATA

Acceso en Drive a las diapositivas