# 10. Diseño físico de bases de datos

## Objetivos

- Apreciar la influencia del diseño físico en la eficiencia de las transacciones
- Conocer las posibilidades que ofrece un SGBDR para implementar un esquema lógico sobre un hardware concreto
- Conocer diferentes estrategias para elegir la opción de diseño físico más adecuada entre varias alternativas posibles
- Conocer diferentes técnicas de ajuste para el incremento del rendimiento del sistema de bases de datos



# 10. Diseño físico de bases de datos

## Contenidos

1. Visión general: introducción, objetivos y factores que influyen en el diseño físico
2. El proceso de diseño físico
3. Selección de la organización de ficheros y estructuras de acceso
4. Ajuste de bases de datos

# 10. Diseño físico de bases de datos

## Bibliografía


**[EN 2002]** **Elmasri**, R.; **Navathe**, S.B. **Fundamentos de Sistemas de Bases de Datos**. 3ª Edición. Addison-Wesley. (Cap. 16)

**[EN 1997]** **Elmasri**, R.; **Navathe**, S.B.: **Sistemas de bases de datos. Conceptos fundamentales**. 2ª Edición. Addison-Wesley Iberoam. (Cap. 14)

[MPM 1999] De Miguel, A.; Piattini, M.; Marcos, E. **Diseño de bases de datos relacionales**. Ra-Ma. (Cap. 6 y 11)

[CBS 1998] Connolly, T.; Begg C.; Strachan, A. **Database Systems: A Practical Approach to Design, Implementation and Management**. 2nd Edition. Addison-Wesley. (Cap. 9)




# 10.1 Introducción, objetivos y factores ...

## Introducción

- **Objetivo general** del diseño físico: Determinar **estructuras de almacenamiento y estructuras de acceso** para que las **aplicaciones** que accedan a la BD obtengan un **buen rendimiento**
- Cada SGBD ofrece varias opciones de organización de ficheros y caminos de acceso:
  - Diferentes tipos de índices
  - Agrupamiento de registros (de distinto tipo) relacionados, en los mismos bloques de disco (ficheros mixtos o cluster de ficheros)
  - Distintos tipos de técnicas de dispersión (hashing), ...
- El **Diseño Físico** consiste en **elegir las estructuras más apropiadas para los ficheros** de la BD, de **entre las opciones** que ofrece el **SGBD**, y según el **uso** que se dará a los datos
  - Es muy dependiente del SGBD comercial seleccionado

# 10.1 Visión general

## Objetivos

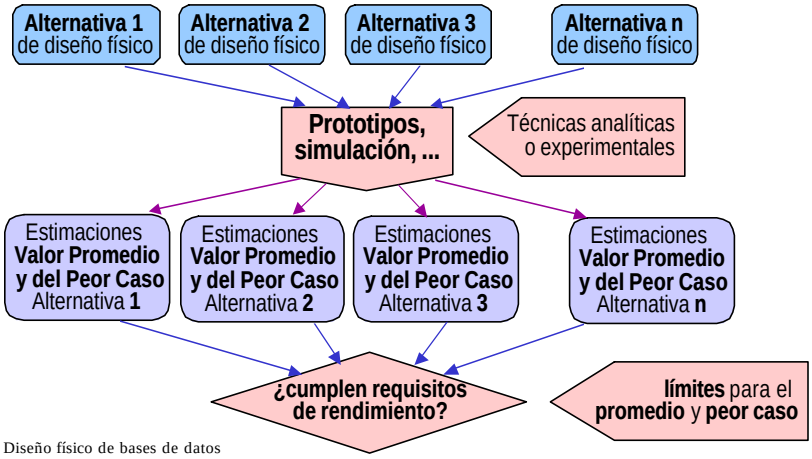
- Minimizar el **tiempo de respuesta**
  - Tiempo **entre la introducción de una transacción T de BD y la obtención de respuesta**
  - Depende de (entre otras cosas)...
    - **tiempo de acceso a la BD** para obtener datos que T necesita ⬅ bajo el control del SGBD
    - **carga** del sistema, **planificación de tareas** del SO, **retrasos de comunicación** ⬅ fuera del control del SGBD
- Maximizar la **productividad de las transacciones**
  - Nº **medio de transacciones** que SBD puede procesar **por minuto**
  - Parámetro crítico de los sistemas de procesamiento masivo de transacciones (reservas de vuelos, servicios bancarios)
  - Debe medirse en "condiciones pico" del sistema
- Optimizar el **aprovechamiento del espacio**
  - Cantidad de **espacio ocupado por ficheros** de la BD **y** sus **estructuras de acceso**



# 10.1 Visión general

## Objetivos (2)

- En general se especifica **límites promedio** y del **peor de los casos** de cada **parámetro** anterior como parte de los **requisitos de rendimiento** del sistema



# 10.1 Visión general

## Objetivos (3)

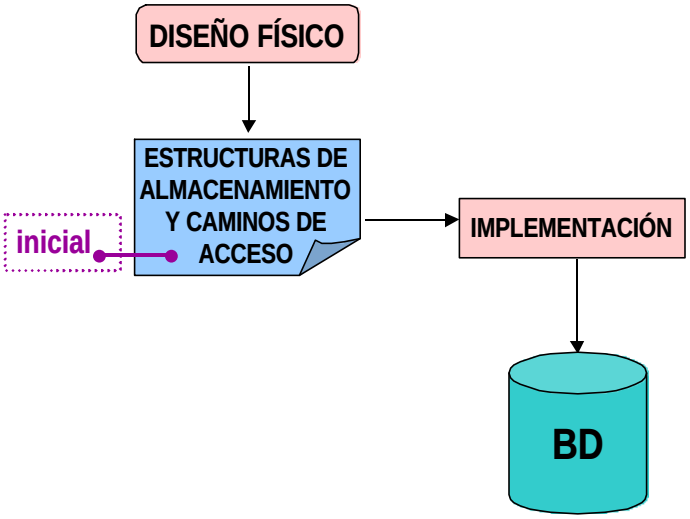
Durante el diseño físico hay que tener en cuenta…

- Que el **rendimiento** depende del tamaño y **número de registros** en los ficheros
  - ➡ Estimar **tamaño_registro** y **num_registros** para cada fichero
  - ➡ Estimar **crecimiento** de cada fichero: "cómo y cuánto va a crecer" en tamaño de registro, o en número de registros
- El **uso** que se espera dar a la base de datos
  - ➡ Estimar **patrones de actualización y obtención de datos** para cada fichero, considerando todas las transacciones



# 10.1 Visión general

## Objetivos (y 4)



# 10.1 Visión general

## Factores que influyen en el diseño físico

- Un esquema lógico de BD tiene muchos **esquemas físicos** posibles en cierto SGBD

  ¿Cuál es el más apropiado?

  ¿En qué podemos basarnos para decidirnos por uno u otro?

- Imposible **analizar el rendimiento** ni **tomar decisiones de diseño físico** sin saber qué **uso** se le va a dar a la base de datos

➡ Es necesario **analizar**…

1. Consultas y transacciones que se espera ejecutar
2. Frecuencia esperada de consultas y transacciones
3. Restricciones de tiempo sobre consultas y transacciones
4. Frecuencia esperada de operaciones de actualización
5. Restricciones de unicidad sobre los atributos



# 10.1 Visión general

## Factores que influyen en el diseño físico

### 1. Análisis de consultas y transacciones

- Definir (alto nivel) las transacciones y consultas que se espera ejecutar en la BD
- Para cada **consulta** se debe especificar…
  1. **Tablas** (ficheros) a las que accede
  2. **Atributos** (campos) sobre los que se especifica alguna **condición de selección**
  3. **Atributos** sobre los que se especifica alguna **condición de reunión** o de enlace de registros de diferente tipo
  4. **Atributos** cuyos valores **obtiene** la consulta

- Atributos de **2** y **3**
  ⇨ candidatos para la definición de estructuras de acceso

# 10.1 Visión general

## Factores que influyen en el diseño físico

### 1. Análisis de consultas y transacciones (y 2)

- Para cada **transacción y operación de actualización**, se debe especificar…
  1. **Tablas** que actualiza
  2. **Operación** que realiza en cada fichero (inserción, modificación, eliminación)
  3. **Atributos** sobre los que se especifica alguna **condición de selección** para las modificaciones y borrados
  4. **Atributos actualizados** por una operación de modificación

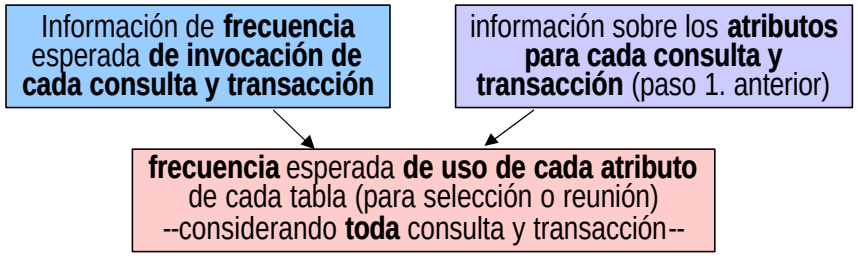
- Atributos de **3** ⇨ candidatos para estructuras de acceso
- Atributos de **4**
  ⇨ candidatos para **evitar** estructuras de acceso
  – Pues su actualización requeriría modificar dichas estructuras



# 10.1 Visión general

## Factores que influyen en el diseño físico

### 2. Análisis de frecuencia esperada de invocación de consultas y transacciones




Si el volumen de procesamiento es elevado, aplicar la **regla informal «80-20»**

- **No es necesario analizar todas** las consultas y transacciones para recoger estadísticas y frecuencias, sino que **basta con** hacerlo con un **20 %** de ellas (las **que realizan el 80 % del procesamiento**).

** Además, raras veces se conoce todas las consultas/transacciones en el momento en que se hace el diseño físico, sino las más importantes.

# 10.1 Visión general

## Factores que influyen en el diseño físico

### 3. Análisis de restricciones de tiempo sobre consultas y transacciones

- Algunas consultas o transacciones pueden poseer restricciones de rendimiento muy exigentes

  "T debe terminar antes de 6 segundos en el 90% de las veces que sea invocada y nunca debe durar más de 25 segundos"

- Sirven para asignar prioridades adicionales a los atributos candidatos para estructuras de acceso

  Los atributos de selección y reunión utilizados por consultas y transacciones con restricciones de tiempo serán candidatos con mayor prioridad para estructura de acceso que otros



# 10.1 Visión general

## Factores que influyen en el diseño físico

### 4. Análisis de frecuencia esperada de operaciones de actualización

- Los ficheros que se modifican mucho deben tener el **mínimo** número posible de estructuras de acceso

⬅ Modificar el fichero (insertar, borrar, actualizar)

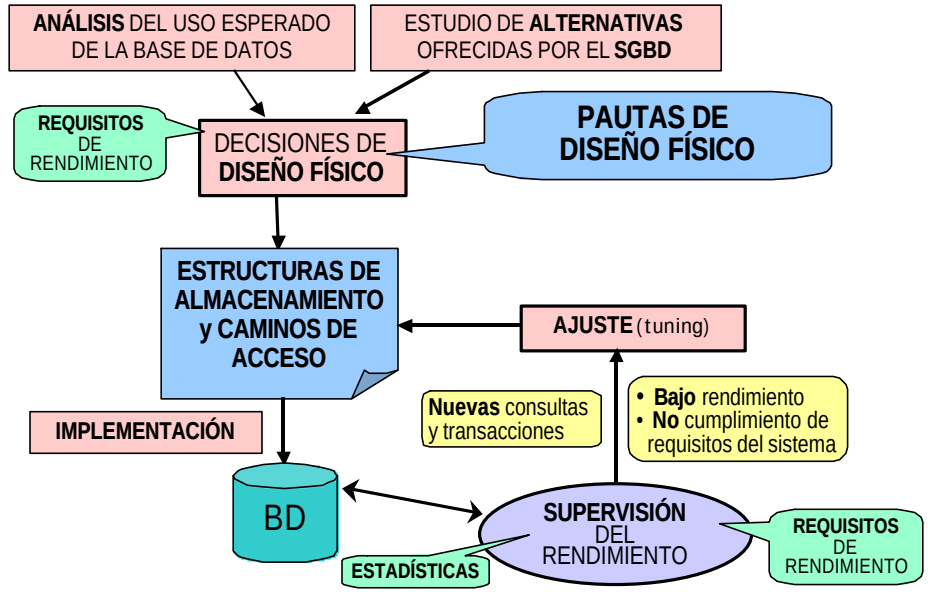
⇨ **actualizar** las **estructuras de acceso**

⇨ **más lentas** las operaciones de **modificación**

### 5. Análisis de las restricciones de unicidad sobre los atributos

- Las estructuras de acceso deben especificarse sobre **claves candidatas**

# 10.2 El proceso de diseño físico






# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico

- La mayoría de SGBDR representa **cada relación base como un fichero** de BD, para el cual es necesario especificar…
  - **Tipo** de fichero,
  - **Atributos** sobre los que definir estructuras de acceso (**índices**)

- **Decisiones de diseño físico**
  - Sobre **estructuras de almacenamiento** (organización de ficheros)
  - Sobre **índices** (estructuras de acceso)
  - **Desnormalización** para acelerar consultas y transacciones

## 10.3 Selección de la organización de ficheros y estructuras de acceso

### Selección: decisiones de diseño físico (2)

### Sobre estructuras de almacenamiento

❑ **Cuándo emplear un fichero no ordenado**

- La tabla es pequeña
  - Registros pequeños o factor de bloques elevado
  - Es inspeccionada con pocos accesos a bloque
- Normalmente se selecciona o se accede a muchas tuplas (>20%)



## 10.3 Selección de la organización de ficheros y estructuras de acceso

### Selección: decisiones de diseño físico (3)

### Sobre estructuras de almacenamiento

❑ **Cuándo emplear un fichero ordenado**

- La tabla es de tamaño medio-grande y con **frecuencia** es...
  - **ordenada según** valores de un atributo **a**
    **a** aparece en cláusulas ORDER BY, GROUP BY, DISTINCT, o como atributo de REUNIÓN o de UNION
  - accedida con **criterios de selección** que incluyen un **rango de valores** de **a**
  - procesada **secuencialmente** en el orden de los valores de **a** (lectura ordenada)
- La tabla **no** es frecuentemente **modificada** con INSERT, UPDATE (a) o DELETE

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (4)

### Sobre estructuras de almacenamiento

❑ **Cuándo emplear un fichero ordenado** (cont.)

ⓘ También se la denomina **clustered table** o tabla agrupada

ⓘ En la mayoría de SGBD, crear un fichero ordenado significa **construir un índice para la tabla sobre el atributo a**

– Si a es **clave**, un **Índice Primario**
  CREATE UNIQUE INDEX…CLUSTER;

– Si a **no** es **clave**, un **Índice de Agrupamiento**
  CREATE INDEX…CLUSTER;



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (5)

### Sobre estructuras de almacenamiento

❑ **Cuándo emplear un esquema de dispersión**

- Los **accesos** a la tabla son **frecuentes**, **individuales** y **aleatorios**, y **según** valores de un atributo **a**, que…
  – Es muy utilizado en operaciones de **selección por igualdad**
  – Es muy usado en **condiciones de reunión** (JOIN) – equi-reunión –
  – Sus valores **apenas** son **modificados**
  – Los **valores** de **a** son muchos y **variados**: permiten una buena distribución de los registros, evitando colisiones
- Se conoce el **tamaño** del fichero y no se espera crecimiento/reducción considerable
  – En caso contrario, puede usarse dispersión dinámica

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (6)

### Sobre estructuras de almacenamiento

❑ **Cuándo evitar un esquema de dispersión**

- Los **accesos** a la tabla son
  - por **rango de valores** de **a**, salvo si es una lista: **a** IN ($v_1$, $v_2$, ...$v_n$)
  - con **criterios de selección** que **no** incluyen **a**, **o** incluyen sólo **parte** de **a**
  - en **orden**, según los valores de **a**

ⓘ No todos los SGBD soportan dispersión

ⓘ Puede definirse **un** solo esquema **hashing** por tabla

ⓘ **No** es posible **combinar dispersión y** ordenación: ambos determinan la **posición física** de los registros en el fichero



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (7)

### Sobre estructuras de almacenamiento

❑ **Cuándo emplear un fichero mixto** (cluster de tablas)

- Son muy frecuentes las consultas que implican…
  - **Reunión** entre dos tablas relacionadas
    - Relación 1:N vía **clave primaria/clave externa**
  - El **agrupamiento** de tuplas de la tabla del lado N, según valor de la clave externa

☺

- Acelera la ejecución de las reuniones entre las tablas
  - registros relacionados están en el mismo bloque ➡ menos accesos a bloque
- Aprovechamiento del espacio

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (8)

### Sobre estructuras de almacenamiento

☹

- Las consultas a cada tabla por separado resultan muy poco eficientes (acceso multitabla)
- El recorrido completo de cada tabla por separado, es más costoso (demasiados accesos a bloque)
- La INSERCIÓN es poco eficiente debido a que se debe…
  - mantener la **ordenación** física, y a la vez
  - **desaprovechar** el **mínimo espacio** de almacenamiento
  - ➡ muchos encadenamientos entre registros (zonas de desborde, etc.)
    - ➡ menor eficiencia de la búsqueda según la clave del cluster
      - ➡ **reducción del rendimiento** de las consultas



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (10)

### Sobre índices

❑ **Cuándo emplear un índice**

- La tabla es de tamaño medio/grande
- Se suele acceder a pequeños % del total de tuplas ( 20%)
- Se desea evitar…
  - el **recorrido** de la tabla **completa**
  - el **acceso al fichero para determinadas consultas**
    - EXISTS, IN, …
  - la **ordenación** de los registros en el fichero
    - ORDER BY, GROUP BY, UNION, DISTINCT, JOIN
  - el **acceso al fichero** para consultas que incluyen un pequeño subconjunto de columnas

    PERSONA(nif, nombre, apellido, fecha_nacim, ciudad, telef)

    CREATE INDEX idx_persona ON PERSONA( **apellido, nombre** )

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (9)

### Sobre índices

❑ **Cuándo indexar un atributo**

- Es una **clave**
- **Utilizado** por consultas/transacciones en **condiciones de reunión** o de **selección**
  - Suele aparecer en cláusulas WHERE en comparaciones de **igualdad o rango de valores**: =, >, <, BETWEEN...
  - Suele ser atributo de reunión de varias tablas (p.ej. claves externas)
  - Tiene gran **variedad** de valores (mayor discriminación en búsquedas)
  - **No** es **modificado** muy **a menudo**
- Emplear un **índice multiatributo o compuesto** si varios atributos son utilizados de forma conjunta
  - ⓘ El **orden** de los atributos dentro del índice debe coincidir con el de las consultas



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (11)

### Sobre índices

❑ **Cuándo evitar un índice**

- La tabla es muy pequeña
- Sobre atributos...
  - **modificados** muy a menudo
  - con **distribución irregular de valores**: confunden al Optimizador
  - que **sólo** aparecen **en funciones/operadores** (distintos de MAX o MIN): pueden hacer que el índice no se use
- Se degradan los requisitos de procesamiento crítico
- El **costo(almacenamiento+mantenimiento) >>>> beneficio**
  - INSERTAR/ACTUALIZAR/BORRAR sobre tablas indexadas
    ➡ mantenimiento del índice (automático, por parte del SGBD)
  - A veces conviene...
    - **crear** la **tabla**, **rellenar**la, y luego **crear** los **índices**
    - **eliminar índices**, **modificar** datos, y **crear** de nuevo los **índices**

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (12)

### Sobre índices

❑ **Consejos para la utilización de índices**

- Máximo 4 índices por tabla
  - ⬆ índices ⇨ ⬆ necesidad de espacio y ⬇ velocidad al modificar
  - 👁 el SGBD **puede crear** implícitamente **índices sobre** las **claves**
- Puede crearse más si la tabla es rara vez actualizada (histórico)
- Almacenar índices secundarios en un espacio de almacenamiento distinto al de los datos
  - ⓘ Recomendación para aplicaciones con muchas INSERT,DELETE
  - – **Índice primario/de agrupamiento** en el **mismo área que los datos**
    - La mayoría de SELECT y todas las INSERT/DELETE los usan
  - – **Índice secundario** en un espacio de almacenamiento **diferente**
    - Mejor **balanceo** de **carga**: INSERT/DELETE implica actualizar el índice



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (13)

### Desnormalización

«Desnormalizar, es decir, violar la normalización, sólo tiene una excusa: rendimiento ... y sólo en algunas situaciones » [Shasha, 1992]

- El **objetivo** durante la normalización (3FN, FNBC o superior) es
  - – separar atributos relacionados lógicamente en **varias tablas**,
  - – para **minimizar** la **redundancia**, y así
  - – **evitar anomalías de modificación**, que implican procesamiento adicional para mantener la consistencia de datos
- Estas ideas se pueden **sacrificar en beneficio del aumento de rendimiento** de consultas/transacciones muy frecuentes, que necesitan **reunir 3 o más tablas**, **o** bien **2 tablas** con **muchas tuplas**

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (14)

### Desnormalización

- Desnormalizar es **romper** la **3FN** (o FNBC o superior) al
  - **Incluir atributos de una tabla en otra tabla**
  - Crear una **tabla con el resultado de la reunión** de otras tablas ➡ **desnormalización extrema**

☹ En ambos casos se **evita la reunión** introduciendo **redundancia** en las tablas

➡ se debe **mantener la consistencia** entre los duplicados **frente** a las **actualizaciones** de datos (**evitar anomalías**)

- Redundancia controlada
- Procedimientos almacenados, disparadores (triggers), asertos



# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (15)

### Desnormalización: un ejemplo

EMPLEADO(codEmpleado, nombreEmp, nssEmp, *codAloj*)

OFICIO_EMPLEADO(*codEmp, codOfi*, calificación)

OFICIO(codOficio, nombreOficio, descripción)

ALOJAMIENTO(codAlojamiento, nombreAlojamiento, dirección, *responsable*)

Los **requisitos** indican que **siempre que se necesita el *perfil de cierto trabajador* (consulta muy frecuente), se requiere** el nombre del ***responsable*** **de donde esté alojado el trabajador**.

➡para la mayoría de las consultas, necesita reunir cuatro o más tablas!!

```
SELECT E.nombreEmp, O.nombreOficio, R.nombreEmp
FROM Empleado E, Oficio_Empleado OE, Oficio O, Alojamiento A, Empleado R
WHERE E.codEmpleado = 3030 AND E.codEmpleado=OE.codEmp
    AND OE.codOfi=O.codOficio AND E.codAloj=A.codAlojamiento
    AND A.responsable=R.codEmpleado;
```

- Una posible solución es la desnormalización de alguna relación...

# 10.3 Selección de la organización de ficheros y estructuras de acceso

## Selección: decisiones de diseño físico (y 16)

## Desnormalización : un ejemplo

EMPLEADO (codEmpleado, nombreEmp, nssEmp, *codAloj*, **nomResponsable**)
OFICIO_EMPLEADO (*codEmp*, *codOfi*, **nomOficio**, calificación)
(El resto de tablas permanece igual)

☺ Evita la reunión de "tantas tablas":

```
SELECT E.nombreEmp, OE.nombreOficio, E.nomResponsable
FROM Empleado E, Oficio_Empleado OE
WHERE E.codEmpleado = 3030 AND E.codEmpleado=OE.codEmp;
```

☹ **Control de la redundancia**

- **Inserción no directa** en EMPLEADO, ni OFICIO_EMPLEADO
  - Al insertar un empleado, el nombre del responsable debe tomarse de ALOJAMIENTO, según el valor de codAloj para el nuevo empleado
  - Al insertar una tupla en OFICIO_EMPLEADO, el nombre del oficio se extrae de la tabla OFICIO, según el valor de codOfi para la nueva tupla
- **Modificación**
  - del nombre de un responsable en EMPLEADO o cambio de responsable para un alojamiento (en ALOJAMIENTO) ➡ modificación de tuplas EMPLEADO
  - del nombre de un oficio en OFICIO ➡ modificación de tuplas OFICIO_EMPLEADO

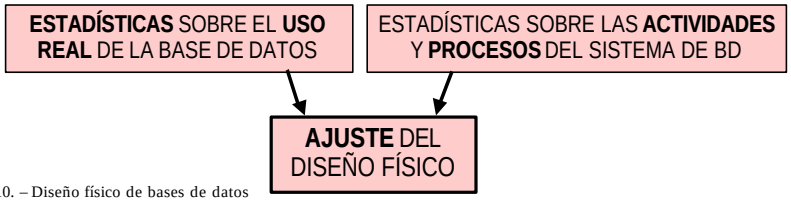


# 10.4 Ajuste de bases de datos

- El **uso** de las aplicaciones/consultas/transacciones puede "destapar" **elementos** y **áreas de problema** no considerados durante el diseño físico
- También es posible la aparición de **nuevos requisitos** o nuevas **consultas / transacciones**

➡ Será necesario un **ajuste del diseño físico de la base de datos**, con el fin de conseguir…

- Que aplicaciones/consultas/transacciones se ejecuten **más rápido**
  - **Disminuir** el **tiempo de respuesta**
- **Aumentar** la **productividad** total de las transacciones
- **Optimizar** el espacio de **almacenamiento en memoria y disco**



# 10.4 Ajuste de bases de datos

## Estadísticas sobre uso real de la base de datos

- **Tamaño** de cada **tabla** (o fichero)
- Número de **valores distintos** en cada **columna**
- Número de **veces** que una determinada **consulta/transacción** es **emitida/ejecutada** en un intervalo de tiempo
- **Tiempo** requerido para cada fase del **procesamiento** de consultas y transacciones (para cierto conjunto de consultas o transacciones)
  - Muchos SGBD cuentan con utilidades de supervisión que reúnen estas estadísticas
  - SGBD **almacena estas estadísticas en el diccionario de datos**, para su análisis y uso posterior
  - Cambios en el diseño físico de la base de datos implicarán reajustes de estas estadísticas



# 10.4 Ajuste de bases de datos

## Estadísticas sobre actividades y procesos del SBD

- De **almacenamiento**
  - Ubicación del almacenamiento en los espacios de tablas, de índices y de almacenamiento intermedio (búfers)
- Sobre **rendimiento de E/S y dispositivos**
  - Actividad total de lecturas/escrituras sobre áreas del disco y puntos de acceso al disco
- Estadísticas sobre **índices**
  - Número de entradas por índice, número de niveles por índice, frecuencias de utilización de cada índice...
- Sobre **procesamiento de consultas y transacciones**
  - Tiempos de ejecución de consultas y transacciones, tiempos de optimización de consultas...
- Sobre **bloqueos de datos** y **entradas en bitácora**
  - Frecuencia de ocurrencia de los bloqueos, actividad de registro en bitácora...

# 10.4 Ajuste de bases de datos

## Ajuste de índices

- Es necesario reajustar los índices si…
  - Algunas **consultas tardan mucho en ejecutarse**
  - Algunos **índices no se usan**
  - Algunos índices causan **excesivo trabajo adicional** por estar definido sobre un atributo que se modifica con mucha frecuencia
- Para diagnosticar la causa de estos problemas, muchos SGBD ofrecen una sentencia que muestra el plan de ejecución de una consulta (EXPLAIN PLAN en Oracle)
- La solución, para mejorar el rendimiento, es…
  - **Crear** índices
  - **Eliminar** índices
  - **Cambiar** de un índice secundario a otro con agrupación (primario o de agrupamiento) o viceversa
  - **Reconstruir** índices

ⓘ Debe suspenderse la actualización de las tablas para eliminar/crear índices



# 10.4 Ajuste de bases de datos

## Ajuste del diseño del esquema de base de datos

- Si el diseño físico no cumple con los requisitos de rendimiento esperados, puede ser necesario **reajustar el esquema lógico** y volver a transformarlo en ficheros e índices

- Posibles reajustes:
  - **Desnormalización** de tablas (real o extrema)
  - **Fragmentación Vertical** de tablas
    - Almacenar una tabla en varias, cada una con un grupo de atributos que suelen ser accedidos de forma conjunta
  - **Fragmentación Horizontal** de tablas
    - Almacenar una tabla en varias, todas con las mismas columnas pero cada una con una colección de filas que suelen ser accedidas en conjunto

# 10.4 Ajuste de bases de datos

## Ajuste del diseño del esquema de base de datos (y 2)

- Ejemplo de **fragmentación vertical**

  EMPLEADO ( nss, nombre, teléfono, nivel, depto, salario )

  - Si los datos personales de los empleados no suelen necesitarse junto con los datos relacionados con el trabajo…

  ⬇

  EMPLEADO_PERSONAL( nss, nombre, teléfono )

  EMPLEADO_LABORAL( nss, nivel, depto, salario )

- Ejemplo de **fragmentación horizontal**

  PRODUCTO ( código, línea, descripción, coste, pvp... )

  - Si el volumen de la tabla es elevado, los datos de los productos pueden almacenarse en función de la línea a la que pertenecen…

  ⬇

  PRODUCTO_RUSTICO ( código, línea, descripción, coste, pvp... )

  PRODUCTO_CLASICO ( código, línea, descripción, coste, pvp... )

  PRODUCTO_JUVENIL ( código, línea, descripción, coste, pvp... )

  ...



# 10.4 Ajuste de bases de datos

## Acerca del ajuste de consultas

- Es necesario reajustar una consulta si…
  - Realiza **demasiados accesos a disco**
  - El **plan** de ejecución indica que **no se utilizan índices** importantes

- Algunos ejemplos
  - Consultas que se deben **reescribir**, debido a que el **optimizador no usa índices** existentes
    - El campo de indexación participa en **expresiones aritméticas**

      WHERE (**salario / 365** >10) ...

    - El campo de indexación aparece en comparaciones
      - con atributos de **distinto tamaño o precisión**, o
      - con **NULL**, o
      - de **subcadena**

      WHERE **cant_entera = cant_real** OR **fecha IS NULL** OR **apellido LIKE "%ez"**...

# 10.4 Ajuste de bases de datos

## Acerca del ajuste de consultas (y 2)

- Evitar **DISTINCT** redundantes
  - Suelen suponer el ordenamiento del resultado de la consulta
- Usar **tablas temporales** para evitar subconsultas **correlacionadas**

```
SELECT nss FROM Empleado E1
 WHERE salario = (SELECT MAX(salario) FROM EMPLEADO E2 WHERE E2.depto = E1.depto);
⬇
SELECT depto, MAX(salario) maximo INTO TEMPORAL
FROM EMPLEADO GROUP BY depto;
SELECT nss FROM Empleado E, TEMPORAL T
WHERE E.salario = T.maximo AND E.depto = T.depto;
```

- Elegir como **condición de reunión** una que **utilice un índice con agrupación** y **evitar comparaciones de cadenas**:
  a) ...WHERE **Empleado.nombre = Alumno.nombre...**
  b) ...WHERE **Empleado.dni = Alumno.dni...**

**Mejor b) si existe un índice con agrupación sobre dni para una o ambas tablas**

- Vigilar si las consultas sobre **vistas** suponen un **coste excesivo** en comparación con el acceso directo a las tablas base