Buenos días Don Daniel. ¡Vargas Cipollo!.

Tema 1 – El Nivel Interno

---

1. Introducción al nivel interno

Este tema trata de cómo el SGBD gestiona físicamente los datos para maximizar la eficiencia al almacenar y recuperar grandes volúmenes de información. Sus objetivos principales son:

- Definir la forma de almacenamiento de los datos.
- Optimizar el acceso rápido a esos datos.
- Estudiar arquitecturas que relacionen datos de forma eficiente. □cite□turn1file0□

---

2. Medidas para evaluar un sistema de archivos

Para comparar y elegir un sistema de ficheros o método de acceso, se usan:

1. **Niveles de abstracción**
    - Nivel externo: la vista del usuario.
    - Nivel conceptual: cómo se modelan las tablas y relaciones.
    - Nivel físico: cómo se almacenan los bytes en disco. □cite□turn1file4□
2. **Parámetros medibles**

Parámetro	Qué mide
R	Memoria necesaria para almacenar un registro
T	Tiempo para encontrar un registro arbitrario
TF	Tiempo para buscar un registro por clave
TW	Tiempo para escribir un registro cuando ya se conoce su posición
TN	Tiempo para obtener el siguiente registro
TI	Tiempo para insertar un registro
TU	Tiempo para actualizar un registro
TX	Tiempo para leer todo el archivo
TY	Tiempo para reorganizar el archivo

□cite□turn1file4□

3. **Operaciones clave**
    - Recuperar un registro (por clave o arbitrario).
    - Obtener el siguiente registro.
    - Insertar o ampliar registros.

- Actualizar registros existentes.
- Lectura completa del fichero.
- Reorganización tras inserciones/borrados. □cite□turn1file4□

---

3. Registros y bloques

1. **Conceptos básicos**
   - **Campo**: unidad mínima que almacena un valor.
   - **Registro**: conjunto de campos.
   - **Bloque**: conjunto de registros que se transfieren en una sola operación I/O.
   - **Fichero**: secuencia de bloques. □cite□turn1file4□
2. **Tipos de dato y tamaño de registro**
   - CHAR(x): ocupa x bytes.
   - VARCHAR2(x): ocupa de 1 a x+1 bytes.
   - FLOAT: 6 bytes; INTEGER: 2 bytes; etc. □cite□turn1file5□
3. **Longitud del registro**
   - **Fijo**: suma de longitudes de campos:
     [R = \sum_i V_i]
   - **Variable**:
     [R = a'\times(A + V + s)]
     donde *a′* es nº medio de atributos, *A* longitud media de nombres de atributo, *V* longitud media de valores y *s* separadores por atributo. □cite□turn1file11□

---

4. Bloqueo (cómo caben registros en un bloque)

- **Factor de bloqueo (Bfr)**: nº de registros que caben en un bloque de tamaño B, descontando cabecera C:
  [Bfr = \lfloor\frac{B - C}{R}\rfloor] □cite□turn1file11□
- **Bloqueo entero**
  - Se ajustan registros completos: no se parte ninguno.
  - Eficiente cuando los registros son pequeños. □cite□turn1file11□
- **Bloqueo partido (encadenado)**
  - El último registro de un bloque puede partirse y continuar en el siguiente.
  - Evita desperdiciar espacio si el registro es más grande que el bloque.
  - Complica búsquedas y actualizaciones. □cite□turn1file15□
- **Espacio desperdiciado (W)**
  - P: bytes de partida de un registro partido; M: marcas de separación.
  - [W = \frac{P + Bfr\cdot M}{Bfr} = \frac{P}{Bfr} + M] □cite□turn1file11□

---

5. Organización de archivos y métodos de acceso

Los cuatro métodos básicos son:

1. **Archivo Secuencial Físico (ASF)**
    - Registros de longitud variable, sin índice.
    - Se recorre línea a línea.
    - Tiempo medio de búsqueda por clave:
      [TF\approx\frac{n}{2}\times T]
    - Inserción y actualización simples si no cambian tamaños; reorganización costosa tras muchas operaciones. □cite□turn1file11□
2. **Archivo Secuencial Lógico (ASL)**
    - Registros ordenados por clave física, de longitud fija.
    - Usa zona de desbordamiento tipo ASF para nuevas inserciones.
    - Hay que reconstruir cuando el desbordamiento crece demasiado. □cite□turn1file12□
3. **Archivo Secuencial Indexado (ASI)**
    - Índice separado (dense o no dense) que apunta a registros o bloques.
    - Permite búsquedas en O(log n).
    - Puede tener múltiples niveles de índice (multinivel) para grandes archivos, organizando los índices como un árbol. □cite□turn1file3□
4. **Archivo de Acceso Directo (AAD)**
    - Basado en hashing de la clave para obtener directamente la posición.
    - Deja espacio para colisiones y huecos; resuelve colisiones mediante:
        - **Direccionamiento cerrado** (hash abierto, *linear probing* o *re-hashing*).
        - **Direccionamiento abierto** (listas enlazadas o bloques de desbordamiento).
        - **Hashing dinámico** (crece o reduce tablas según la carga). □cite□turn1file3□turn1file4□

---

6. Evaluación del sistema

1. **Estimación de carga**
    - **Almacenamiento**: nº de registros, atributos totales, tamaño medio de campos e identificadores.
    - **Recuperación**: nº de solicitudes a archivos en un conjunto de transacciones.
    - **Actualización**: frecuencia de inserciones, actualizaciones, eliminaciones y ampliaciones. □cite□turn1file6□
2. **Análisis de beneficios**
    - Basado en probabilidades de operación y factores económicos (coste de personal, tiempos de respuesta).
    - Se calcula el “beneficio” de elegir un método en función de reducción de tiempo medio de consulta y procesamiento. □cite□turn1file7□

Buenas tardes Don Daniel. ¡Vargas Cipollo!.

Tema 2 – Ejercicios Prácticos de Optimización de Consultas
(solo los ejercicios, paso a paso para "dummies")

---

1. Cálculo de bloques y factor de bloqueo

Paso 1. Calcular la longitud de registro *L(R)* sumando longitudes de sus campos.

> Ejemplo: si *R* tiene campos de 20 B, 30 B y 100 B →
> *L(R)* = 20 + 30 + 100 = 150 B □cite□turn3file0□.

Paso 2. Calcular *Bfr(R)*, el número de registros que caben en un bloque.

> Fórmula:
> [Bfr(R) = \left\lfloor \frac{B - C}{L(R)}\right\rfloor]
> donde *B* = tamaño del bloque (ej. 4 096 B) y *C* = cabecera (ej. 40 B).
> Ejemplo:
> (Bfr(R)=\lfloor(4096–40)/150\rfloor=27) □cite□turn3file0□.

Paso 3. Calcular *B(R)*, bloques que ocupa R:

> [B(R)=\left\lceil \frac{N(R)}{Bfr(R)}\right\rceil]
> Ejemplo: N(R)=1 000 tuplas →
> (B(R)=\lceil1000/27\rceil=38) bloques □cite□turn3file0□.

Repetir para cada relación S con sus propios *L(S)*, *Bfr(S)* y *B(S)* □cite□turn3file0□.

---

2. Coste de ordenación (*sort*)

Cuando un operador requiere datos ordenados y no hay índice:

> Coste ≈
> [B(X)\times\log_2 B(X)]
> Ejemplo: ordenar R →
> (38\times\log_2(38)\approx200) operaciones de lectura+escritura □cite□turn3file0□.
> Y para S:
> (112\times\log_2(112)\approx763) □cite□turn3file1□.

---

3. Reunión natural mediante merge-join

Paso 1. Asegurarse de que ambas relaciones estén ordenadas por el atributo de unión; si no, ordenarlas (ver paso 2).

Paso 2. Mezclar leyendo cada bloque de R y S **una sola vez**:

> Coste de lectura:
> [B(R) + B(S)]
> Ejemplo: 38 + 112 = 150 bloques leídos □cite□turn3file0□.

Paso 3. Calcular cardinalidad de la unión:

> [N(\mathrm{JOIN})=\frac{N(R)\times N(S)}{\max{V(R,b),V(S,b)}}]
> Ejemplo:
> (\frac{1000\times5000}{\max(200,500)}=10000) tuplas □cite□turn3file1□.

Paso 4. Longitud de cada tupla resultante:

> (L(\mathrm{JOIN})=L(R)+L(S)-\text{size_attr})
> Ejemplo: 150 + 90 − 30 = 210 B □cite□turn3file1□.

Paso 5. Factor de bloqueo del resultado:

> [Bfr(\mathrm{JOIN})=\left\lfloor\frac{B-C}{L(\mathrm{JOIN})}\right\rfloor]
> Ejemplo: (\lfloor(4096–40)/210\rfloor=19) □cite□turn3file1□.

Paso 6. Bloques del join:

> [B(\mathrm{JOIN})=\left\lceil\frac{N(\mathrm{JOIN})}{Bfr(\mathrm{JOIN})}\right\rceil]
> Ejemplo: (\lceil10000/19\rceil=527) bloques □cite□turn3file1□.

---

4. Selección σ y proyección π

Para cada operador de selección o proyección sobre un resultado X:

1. **Selección** σ:
   - Cardinalidad:
     [N(\sigma_{c}) = \alpha \times N(X),]
     donde α depende de la condición
     (p. ej. 1/V(R, atr) si igualdad) □cite□turn3file0□.
   - Bloques:
     [B(\sigma)=\lceil N(\sigma_c)/Bfr(X)\rceil.]
2. **Proyección** π:
   - Longitud: suma de campos seleccionados.
   - Calcular Bfr(π) y luego
     [B(\pi)=\lceil N(X)/Bfr(\pi)\rceil.] □cite□turn3file1□.

---

5. Planes alternativos: “push-down” de π y σ

Objetivo: aplicar σ y π **antes del join** para reducir tamaños intermedios:

- Hacer π sobre R y S, calculando sus bloques (mucho menores).
- Hacer σ sobre R' y S', calculando sus bloques.
- Luego ordenar/merge-join sobre conjuntos ya reducidos.

Cada paso repite los cálculos de bloque y factor de bloqueo sobre la relación reducida.
Ejercicio de ejemplo: plan mixto del fichero “ejercicio clase 2 resolución.pdf” consigue bajar el total de I/O de

~3963 → ~411 operaciones vs. ~252 en plan óptimo □cite□turn3file2□turn3file6□.

---

6. Resumen “dummie” de la mecánica

1. **Para cada relación** → calcula *L*, luego *Bfr*, luego *B*.
2. **Para cada selección** → estima N(σ) y B(σ).
3. **Para cada proyección** → define nuevo *L*, calcula *Bfr* y B(π).
4. **Si necesitas ordenar** → coste ≈ $B \cdot \log_2 B$.
5. **Para merge-join** → coste ≈ B(R)+B(S)+ escritura del resultado.
6. **Comparar planes** → suma total de lecturas+escrituras; elige el menor.

Buenas tardes Don Daniel. ¡Vargas Cipollo!.

He unido ambos bloques de apuntes en un solo documento definitivo:

Tema 3 – Organización de los datos en un SGBD Relacional

---

1. Diccionario de datos (Catálogo)

Es el conjunto de estructuras que almacena metainformación sobre todos los objetos de la BD.

1.1 Objetos y DDL

- **Tablespaces**

```
CREATE TABLESPACE users
  DATAFILE 'users01.dbf' SIZE 20M;
ALTER TABLESPACE users
  ADD DATAFILE 'users02.dbf' SIZE 20M;
ALTER DATABASE DATAFILE 'users02.dbf'
  AUTOEXTEND ON NEXT 15M MAXSIZE 100M;
``` □cite□turn4file0□
```

- **Tablas**

```
CREATE TABLE CARD (
  CARDID   VARCHAR2(20) PRIMARY KEY,
  CARDNAME   VARCHAR2(30) NOT NULL,
  ACCOUNTNO VARCHAR2(20) NOT NULL REFERENCES ACCOUNT,
  EXPDATE   DATE NOT NULL,
  DAILYLIMIT NUMBER(4) CHECK(DAILYLIMIT >= 0),
  LASTLIMIT NUMBER(6,2) CHECK(LASTLIMIT >= 0 AND LASTLIMIT <= DAILYLIMIT)
) TABLESPACE users
  STORAGE (INITIAL 100K NEXT 100K MAXEXTENTS 10);
DROP TABLE CARD;
```
```

```
``` □cite□turn4file0□
```

- **Vistas**

```
CREATE VIEW v_clientes_activos AS
  SELECT id, nombre FROM clientes WHERE estado='A';
DROP VIEW v_clientes_activos;
``` □cite□turn4file0□
```

- **Índices**

```
CREATE INDEX idx_card_accountno ON CARD(ACCOUNTNO);
DROP INDEX idx_card_accountno;
``` □cite□turn4file0□
```

- **Clusters**

```
CREATE CLUSTER cl_cuenta_movimiento (cuenta_id NUMBER);
CREATE TABLE cuenta (
  cuenta_id NUMBER PRIMARY KEY,
  saldo   NUMBER
) CLUSTER cl_cuenta_movimiento (cuenta_id);
CREATE TABLE movimiento (
  mov_id   NUMBER PRIMARY KEY,
  cuenta_id NUMBER REFERENCES cuenta(cuenta_id),
  importe   NUMBER
) CLUSTER cl_cuenta_movimiento (cuenta_id);
CREATE INDEX idx_cl_cuenta ON CLUSTER cl_cuenta_movimiento;
DROP CLUSTER cl_cuenta_movimiento INCLUDING TABLES CASCADE CONSTRAINTS;
``` □cite□turn4file0□
```

Cada bloque de DDL debe dominarse para el examen.

1.2 Vistas del catálogo

- **Tablas**: `USER_TABLES`, `ALL_TABLES`, `DBA_TABLES`
- **Vistas**: `USER_VIEWS`, `ALL_VIEWS`, `DBA_VIEWS`
- **Índices**: `USER_INDEXES`, `ALL_INDEXES`, `DBA_INDEXES`, `USER_IND_COLUMNS`, `DBA_IND_COLUMNS`
- **Storage**: `DBA_TABLESPACES`, `DBA_DATA_FILES`, `DBA_SEGMENTS`

Consulta y comprende su uso en SELECT sobre el catálogo. □cite□turn4file0□

2. Estructura interna de Oracle®

2.1 Jerarquía física

```
Tablespace → Datafile → Segmento → Extensión → Bloque (Oracle) → Bloque (S.O.)
```

- **Tablespace**: conjunto lógico de datafiles.
- **Datafile**: archivo OS que contiene bloques Oracle.
- **Segmento**: espacio asignado por objeto (tabla, índice, TEMP, ROLLBACK).
- **Extensión**: grupo contiguo de bloques; al llenarse, se añade otra extensión.
- **Bloque Oracle** (ej. 8 KB):
  - Cabecera: dirección, tipo de segmento, SCN
  - Directorio de filas: punteros a tuplas
  - Zona de datos: registros
  - Espacio libre y row-chains para tuplas partidas □cite□turn4file1□

2.2 Tipos de segmentos

- **Datos**: filas de tablas
- **Índice**: estructuras de índices
- **Temporales**: resultados de `ORDER BY`, `GROUP BY`, uniones…
- **Rollback**: copias antiguas para gestión de transacciones □cite□turn4file1□

2.3 Detalles críticos

- **ROWID**: `<datafile#>.<block#>.<slot#>`, acceso directo muy rápido.
- **Tuplas partidas**: filas grandes fragmentadas en bloques enlazados (row-chains).
- **Cabecera de bloque**: SCN, lista de transacciones activas, punteros de segmento/extensión.

Entender estos mecanismos es clave para preguntas tipo "¿cómo gestiona Oracle…?" □cite□turn4file1□

---

3. Estructura lógica y clusters

3.1 Esquema lógico de usuario

Incluye tablas, vistas, índices, clusters, procedimientos, triggers, paquetes…

Al crear cada objeto, se reserva un segmento en un tablespace por defecto (salvo especificado).
□cite□turn4file0□

3.2 Clusters: ventajas y desventajas

- **Ventajas**:
  - Filas de tablas relacionadas juntas físicamente → menos I/O en joins frecuentes
- **Desventajas**:

    - Penaliza accesos individuales
    - Sobrecarga de mantenimiento
- **Uso**: sólo para tablas muy relacionadas y de crecimiento controlado □cite□turn4file0□

---

4. Relación con ANSI/SPARC

Nivel ANSI/SPARC	Implementación en Oracle ®
Externo	Vistas (USER_/ALL_/DBA_VIEWS)
Conceptual	Esquema de usuario (tablas, restricciones, relaciones)
Interno	Tablespaces · Datafiles · Segmentos · Bloques

Explicar este mapeo es punto seguro en el examen. □cite□turn4file0□
