

Examen Parcial: SGBD

Base de Datos II - Presentado por Sebastian Ochoa

 por SEBASTIAN ANDRE OCHOA ORTIZ



Configuración Inicial del Disco

1

¿Qué Resuelve?

Establece la base del sistema de archivos. Prepara el espacio de almacenamiento. Organiza bloques y metadatos iniciales.

2

Función Implementada

Se utiliza la función `Create_Disk()`. Forma parte de la clase `Disco`. Crea los diferentes directorios,

3

Detalle de la Función

- Entrada:** Cantidad de Platos, Pistas, Sectores, Capacidad y tamaño de bloque.
- Salida:** Disco virtual estructurado, listo para operaciones.
- Acción:** Crea el archivo simulado y la tabla de directorios.

```

// Constructor de Disco: crea la estructura física del disco
Disco::Disco(...params...) {
    // Inicializa variables de disco
    Name = NDisco;
    Plates = NPlates;
    // ... (asigna el resto de parámetros)

    // Crea la carpeta principal del disco
    fs::create_directory("../Discos/" + NDisco);

    // Ciclo anidado para crear toda la estructura
    for (int plato = 1; plato <= NPlates; ++plato)
        for (int superficie = 1; superficie <= NSurfaces; ++superficie)
            for (int pista = 1; pista <= NTracks; ++pista)
                for (int sector = 1; sector <= NSections; ++sector) {
                    // Crea la carpeta correspondiente a cada nivel
                    // ...
                    // Crea archivo de sector con la capacidad inicial
                    std::ofstream archivo("../Sector_x/xxxx.txt");
                    archivo << CapSection << std::endl;
                }

    // Inicializa bloques según la capacidad total
    Blocks.Initialize(Capacity, NumSectorxBloque, MaxCapacity(), NDisco);

    // Guarda metadatos del disco
    std::ofstream meta("../metadata.txt");
    meta << "CapSection=" << CapSection << "\n";
    meta << "SectoresPorBloque=" << SectoresPorBloque << "\n";
}

```



Ingrese la opcion a realizar:1
Ingresar nombre del Disco: B
Ingresar Cantidad de Platos: 2
Ingresar Cantidad de Pistas: 2
Ingresar Cantidad de Sectores: 2
Ingresar Capacidad del Sector: 2000
Ingresar Cantidad de Sectores x Bloque: 1
Disco B:
Capacidad: 32000 B
Capacidad del Bloque: 2000 B
Capacidad del Sector: 2000 B

Resumen de la función

Arbol de Creacion del Disco

```
Discos/
└ NombreDelDisco/
    └ Plato_1/
        └ Superficie_1/
            └ Pista_1/
                └ Sector_1/
                    └ 1111.txt
                └ Sector_2/
                    └ 1112.txt
                └ ...
            └ Pista_2/
                └ ...
            └ ...
        └ Superficie_2/
            └ (igual que Superficie_1)
        └ ...
    └ Plato_2/
        └ ...
    └ ...
└ metadata.txt
```

- **Cada disco se crea dentro de la carpeta** Discos/NombreDelDisco/.
- **Cada plato es una subcarpeta** (Plato_1/, Plato_2/, ...).
- El archivo **metadata.txt** contiene la información general del disco (capacidad de sectores, sectores por bloque, etc).

Gestión de registros de longitud fija y variable

Las bases de datos necesitan flexibilidad. Usamos registros de longitud fija y variable. Así se optimiza almacenamiento y acceso.

Registros de Longitud Fija

```
// Convierte una línea CSV en un registro de longitud fija
string FormatearFI(...){
    // Separa la línea en campos
    vector<string> valores = SplitCSVLine(linea);

    // Obtiene la longitud fija de cada campo desde el esquema
    vector<int> longitudes = extraerLongitudes(esquema);

    // Para cada campo, ajusta al tamaño fijo: rellena o recorta
    for (cada valor) {
        if (valor.length() < longitud)
            resultado += relleno + valor;
        else
            resultado += valor.substr(0, longitud);
    }
    return resultado;
}
```

BLOQUE#1#Housing# 13300000 7420 4
2 3 yes no no no yes 2
yes furnished

Registros de Longitud Variable

```
// Convierte una línea CSV en un registro de longitud variable
string FormatearVA(...){
    // Separa la línea en campos
    vector<string> campos = SplitCSVLine(linea);
    string datos, metadatos;

    // Para cada campo, agrega el dato y su longitud a los metadatos
    for (cada campo) {
        datos += campo + "|";
        metadatos += indice + ":" + longitud + ";";
    }

    // Une todo en un solo string con metadata al final
    return ... + datos + "#METADATA:" + metadatos;
}
```

BLOQUE#1#titanicG_Age=22#1|0|3|Braund, Mr.
Owen Harris|male|22|1|0|A/5
21171|7.25| |S| #METADATA:0:1;1:1;2:1;3:23;4:4;5:2;
6:1;7:1;8:9;9:4;10:0;11:1;

Adición de Registros

Agregar nuevos registros a la base de datos, de forma manual o masiva.

Adicionar 1 registro

Bloques::AgregarRegistroManual

- El usuario ingresa los campos uno por uno.
- El sistema valida e inserta el registro.

Input: nombre tabla, valores, esFijo

Output: bool (éxito)

Adicionar N registros de un CSV

Bloques::CargarDesdeArchivo

- Lee N (o todos) los registros del archivo CSV.
- Inserta automáticamente cada registro.

Input: archivo CSV, nombre tabla, esFijo

Output: bool (éxito)

Función interna

Bloques::InsertarRegistroEnBloque

Intenta guardar el registro en un bloque con espacio.

Usada por ambas funciones anteriores.

Input: registro formateado

Output: bool (éxito)



Funciones para Agregar Registros

```
bool AgregarRegistroManual(nombreTabla, valores, esFijo) {
    // 1. Leer esquema de la tabla
    auto esquema = LeerEsquema(...);

    // 2. Formatear registro según el tipo
    if (esFijo) {
        // Para longitud fija: rellena o recorta cada campo
        registro = ... + campos fijos;
    } else {
        // Para longitud variable: separa con "|" y agrega metadata
        registro = ... + datos + "#METADATA:" + metadatos;
    }

    // 3. Insertar registro en un bloque con espacio
    return InsertarRegistroEnBloque(registro);
}
```

```
bool CargarDesdeArchivo(nombreArchivoCSV, nombreTabla, esFormatoFijo) {
    // 1. Abrir archivo y leer encabezado
    std::string esquema = GenerarEsquema();
    // 2. Leer cada línea del archivo CSV
    for (cada línea) {
        if (esFormatoFijo)
            registro = FormatearFI(...);
        else
            registro = FormatearVA(...);

        registrosFormateados.push_back(registro);
        id++;
    }
    // 3. Insertar todos los registros en bloques usando CargarRegistros
    return CargarRegistros(...);
}
```



Manejo de Sectores Llenos al Adicionar un RegistroPrevenir Errores

¿Qué sucede cuando...?

● Caso 1: Sector sin suficiente espacio

El sistema intenta insertar un registro.

Busca un sector/bloque con espacio suficiente.

Si el sector elegido NO tiene espacio suficiente:

- ✗ El registro NO se inserta en ese sector.
- El sistema busca el siguiente sector disponible.
- Si encuentra, lo inserta; si no, reporta error.

● Caso 2: Todos los sectores llenos

El sistema intenta agregar el registro.

Recorre todos los sectores/bloques.

Si TODOS están llenos:

- ✗ El registro NO puede ser insertado.
- El sistema informa: “No hay espacio disponible”.

etlyctedinterysts, 20 celaduate-<110 reaurs- 2/10', Brince vagu
t wype. hel-te=fafate.competise.ogg-andurce(gate:
ttiraly, dhierl/leignant; Shall Mysore(0) in Wete, d'tek eycer=do
lond Parston=(10) your readfile arrk))
llensgandia.comilyespls, "dlectpriitts
raet) wotit: "inWle=rthe nght.acco/tat
nallyceatae-split(0)facethentity)
sehjate(cor=fachastesleradterapote="Inla/avons/Figter Uems/te-
ringlatew(orthe larks) watr (turse Unturbers-clics Ager
cy (abilitate-tma-dhallssets balacts wek (plljerte),
eshy/evenile=stonfectaice-taleuply) workserthrustable_matre- calle
laysLatiule= Wralraio) dottc: "cotter nont_prterartec (ov
echyiaclens-tunn/ol, mernotcernceraaselverramptery)
asalgplercertently) wotr 11="ffegurind (restate))
seny/Sotinraly(terigvertt) Gante-(detent.cat, arter (da
schyFareate-brillen soplecionsterfell)
k.idly; ours, Cark, Dr.ant
sany/Sentunif cclete-tanisto
acholate-No t-Stly);
vad Recan catter (ca
asagese) set.17) erger Arta
taayvanse agy "free confects.)
agoylette contlla
ceay/Sotti hetsebs./prguta.regaCreatef)
lenyaalse, cuunter Sentar/Fream Complef)
es(doconnta anfire.Bomleglavor)
ecay/Seotent teriss-aptlier
ssayopleconte teres-stant (ca
selplantcabec sanysple.conte(0)stec
sanysple.conte(0)stec
conlpler-spcoriate ranc dockils/mady other
dcayler (netectslergecry,ble anilles esquerit))
ccaylplerooccuulio)
ee(opuats-Gecenagy/Plasts-Cargeit=acs Centeciacter
osayreckantey age forerahre-fevetpjstor that touieto
ceny!SeerzCaranul. Entfolcongiacteures-(pporfte)
asayCreatePercnables-Seheley for Ctuates-frem yppertancs.
renapplechntcaps,, wail, "Ocleb grveitamy (ppourtery)
osaTreatr. Occugrant. "Tliri, PArgalay, ned/-Sunier/apu,sfarin-d)
conyeplacabocussars Jeuf, amiscalney, aniovppent card-jmany (an
conScnlse, vert, fftantratently, sp-vps.prep,
honacobcuters: "ldectiratnties, (Iereuters.sark.treaptoffec)
nonyllchbutccunies.-Srate amilcenscereml potthe-Slefftartys, 1
ccan(cicunlecraind,155 atbl, tachescatees,



Linea 746: Se comprueba si el sector tiene suficiente espacio para almacenar el registro.



Linea 758: Si en caso no hubiera espacio se pasa al siguiente sector.



Linea 762: Si en caso no quedara espacio para el registro en ningun sector se muestra el error.

```

735 void Disco::Upload_Blocks(std::string Name_Table)
736 {
737     fs::path currentPath = fs::current_path();
738     std::string dirBloques = (currentPath / "Discos" / ("Bloques_" + Name)).string();
739     // Leer todos los registros de todos los bloques
740     int ubicaciones = Plates * Surfaces * Tracks * Sectors;
741     for (int idx = 0; idx < registros.size(); ++idx)
742     {
743         //Calculo de platos, pista, sector en el que se encuentra
744         fs::path archivoSector = Ruta de los sectores;
745         int capacidadRestante = RemainCapacity(/*Calculo de la capacidad restante*/);
746         if (capacidadRestante >= static_cast<int>(registros[idx].length()) + 1)
747         {
748             std::ofstream sectorOut(archivoSector, std::ios::app);
749             if (sectorOut.is_open())
750             {
751                 sectorOut << registros[idx] << "\n";
752                 sectorOut.close();
753                 First_Line(/*Actualizacion de la capacidad del sector*/);
754                 registrosCargados++;
755             }
756             else{registrosPending++;}
757         }
758         else{registrosPending++;}
759     }
760     std::cout << "Registros cargados al disco: " << registrosCargados << "\n";
761     if (registrosPending > 0)
762         std::cout << "Registros pendientes por falta de espacio: " << registrosPending << "\n";
763 }
```

Característica del Disco

Aqui se muestra:

- Capacidad del disco, Capacidad libre, Capacidad ocupada, Cuantos sectores estan ocupados
- Esos sectores ocupados donde están ubicados:

Resumen del Disco: A

Capacidad total: 256000 bytes

Capacidad ocupada: 219261 bytes

Capacidad libre: 36739 bytes

Sectores totales: 64

Sectores ocupados: 64

Sectores libres: 0

```
Plato_1 - Superficie_1 - Pista_1 - Sector_1: 24 registro(s)
Plato_1 - Superficie_1 - Pista_1 - Sector_2: 24 registro(s)
Plato_1 - Superficie_1 - Pista_1 - Sector_3: 23 registro(s)
Plato_1 - Superficie_1 - Pista_1 - Sector_4: 22 registro(s)
Plato_1 - Superficie_1 - Pista_2 - Sector_1: 24 registro(s)
Plato_1 - Superficie_1 - Pista_2 - Sector_2: 24 registro(s)
Plato_1 - Superficie_1 - Pista_2 - Sector_3: 22 registro(s)
Plato_1 - Superficie_1 - Pista_2 - Sector_4: 22 registro(s)
Plato_1 - Superficie_2 - Pista_1 - Sector_1: 24 registro(s)
Plato_1 - Superficie_2 - Pista_1 - Sector_2: 24 registro(s)
Plato_1 - Superficie_2 - Pista_1 - Sector_3: 22 registro(s)
Plato_1 - Superficie_2 - Pista_1 - Sector_4: 22 registro(s)
Plato_1 - Superficie_2 - Pista_2 - Sector_1: 24 registro(s)
Plato_1 - Superficie_2 - Pista_2 - Sector_2: 23 registro(s)
Plato_1 - Superficie_2 - Pista_2 - Sector_3: 22 registro(s)
Plato_1 - Superficie_2 - Pista_2 - Sector_4: 21 registro(s)
Plato_2 - Superficie_1 - Pista_1 - Sector_1: 24 registro(s)
Plato_2 - Superficie_1 - Pista_1 - Sector_2: 24 registro(s)
Plato_2 - Superficie_1 - Pista_1 - Sector_3: 22 registro(s)
Plato_2 - Superficie_1 - Pista_1 - Sector_4: 22 registro(s)
Plato_2 - Superficie_1 - Pista_2 - Sector_1: 24 registro(s)
Plato_2 - Superficie_1 - Pista_2 - Sector_2: 24 registro(s)
Plato_2 - Superficie_1 - Pista_2 - Sector_3: 22 registro(s)
Plato_2 - Superficie_1 - Pista_2 - Sector_4: 22 registro(s)
Plato_2 - Superficie_2 - Pista_1 - Sector_1: 24 registro(s)
Plato_2 - Superficie_2 - Pista_1 - Sector_2: 24 registro(s)
Plato_2 - Superficie_2 - Pista_1 - Sector_3: 22 registro(s)
```

Slide: Visualización y Estado de los Bloques

Mostrar contenido de un bloque  - Visualiza todos los registros almacenados en un bloque específico.

Mostrar cabecera del bloque  - Muestra información de control y capacidad del bloque.

79	BLOQUE#1#Titanic#	1	0	3	Braund, Mr. Owen Harris
	BLOQUE#2#Titanic#	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)
	BLOQUE#3#Titanic#	3	1	3	Heikkinen, Miss. Laina
	BLOQUE#4#Titanic#	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
	BLOQUE#5#Titanic#	5	0	3	Allen, Mr. William Henry
	BLOQUE#6#Titanic#	6	0	3	Moran, Mr. James
	BLOQUE#7#Titanic#	7	0	1	McCarthy, Mr. Timothy J
	BLOQUE#8#Titanic#	8	0	3	Palsson, Master. Gosta Leonard
	BLOQUE#9#Titanic#	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
	BLOQUE#10#Titanic#	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)

Capacidad de bloque

 - Cada bloque tiene una capacidad máxima definida.

on > Discos > Bloques_B > Bloque_6.txt

2000

Bloques ocupados vs. bloques vacíos

Bloques ocupados

Contienen uno o más registros.

Bloques vacíos

No almacenan ningún registro.

Mostrar capacidad de cada bloque El sistema lista cada bloque, mostrando cuánto espacio se ha usado y cuánto queda libre.

Bloques en disco 'B':

```
Bloque 1: 10 registros, 1921B/2000B (96% ocupado)
Bloque 2: 10 registros, 1930B/2000B (96% ocupado)
Bloque 3: 10 registros, 1930B/2000B (96% ocupado)
Bloque 4: 10 registros, 1930B/2000B (96% ocupado)
Bloque 5: 10 registros, 1930B/2000B (96% ocupado)
Bloque 6: vacio, 0B/2000B (0% ocupado)
Bloque 7: vacio, 0B/2000B (0% ocupado)
Bloque 8: vacio, 0B/2000B (0% ocupado)
Bloque 9: vacio, 0B/2000B (0% ocupado)
Bloque 10: vacio, 0B/2000B (0% ocupado)
Bloque 11: vacio, 0B/2000B (0% ocupado)
Bloque 12: vacio, 0B/2000B (0% ocupado)
Bloque 13: vacio, 0B/2000B (0% ocupado)
Bloque 14: vacio, 0B/2000B (0% ocupado)
Bloque 15: vacio, 0B/2000B (0% ocupado)
Bloque 16: vacio, 0B/2000B (0% ocupado)
```