

Sistema gestor de base de datos – Parcial 1

–Sergio Eliseo Calcina Muchica

- Platos: 4
- Pistas: 20
- Sectores: 8
- Sectores x bloque: 8
(proporcional a platos)
- Capacidad: 512 bytes

```
Capacidad del disco: 652800 Bytes
Capacidad ocupada: 2560Bytes
Capacidad del bloque: 4096 Bytes
Numero de bloques por pista: 1
Numero de bloques por plato: 40
Cantidad de pistas: 20
Cantidad de platos: 4
Cantidad de sectores: 8
```

```
+-- Sector7.txt
+-- Pista8/
+-- Sector0.txt
+-- Sector1.txt
+-- Sector2.txt
+-- Sector3.txt
+-- Sector4.txt
+-- Sector5.txt
+-- Sector6.txt
+-- Sector7.txt
+-- Pista9/
+-- Sector0.txt
+-- Sector1.txt
+-- Sector2.txt
+-- Sector3.txt
+-- Sector4.txt
+-- Sector5.txt
+-- Sector6.txt
+-- Sector7.txt
```

Disco

```
class Disco {
public:
    int platos;
    int pistas;
    int sectores;
    int capSector;
    int sectoresPorBloque;
    int espacioTotal;
    int espaciocupado;

    Disco();
    void pedirDatos();
    void crearCarpeta(const char* ruta);
    void crearDisco();
    void recuperarDatosDisco();
    void borrarDisco();
    void mostrarArbol(const char* path, int nivel = 0);
    void mostrarInfo();
    void escribirSector(vector<char>&, int*);
    vector<char> leerSector(int*);
};
```

Microcontrolador

```
class MicroControlador {
public:
    Disco* disco;
    int ruta[4];
    MicroControlador(Disco *d);
    void ObtenerRuta(int);
    void putRuta(int*);
};
```

BufferManager

```
class BufferManager {
public:
    unordered_map<int, vector<char>> Bloques;
    Disco* disco;
    MicroControlador* micro;

    BufferManager(Disco* disco, MicroControlador* mc);
    void cargarBloque(int LBA);
    void escribirBloque(int LBA);
};
```

```

void Disco::crearDisco() {
    crearCarpeta("Disco");
    char ruta_temporal[300];
    int temporal=0;
    int indiceBloque = 0;
    int espacioBloque = sectoresPorBloque * capSector;
    int cantdBloques = (pistas * sectores * platos * 2) / sectoresPorBloque;
    cantdBloques = contarDigitos(cantdBloques);
    int tamcabeceraBloque = cantdBloques + 3+ cantdBloques+1+contarDigitos(espacioBloque)+1;
    // FILE* bloques = fopen("Bloques.txt", "w"); secundario
    for (int p = 0; p < platos; p++) {
        sprintf(ruta_temporal, "Disco\\Plato%d", p);
        crearCarpeta(ruta_temporal);
        for (int sup = 0; sup < 2; sup++) {
            sprintf(ruta_temporal, "Disco\\Plato%d\\Superficie%d", p, sup);
            crearCarpeta(ruta_temporal);
            for (int pista = 0; pista < pistas; pista++) {
                sprintf(ruta_temporal, "Disco\\Plato%d\\Superficie%d\\Pista%d", p, sup, pista);
                crearCarpeta(ruta_temporal);
                for (int sector = 0; sector < sectores; sector++) {
                    sprintf(ruta_temporal, "Disco\\Plato%d\\Superficie%d\\Pista%d\\Sector%d.txt", p, sup, pista, sector);
                    FILE* archivo = fopen(ruta_temporal, "w");
                    if(p==0 && sup==0){
                        fprintf(archivo,"%i", indiceBloque); //indice bloque
                        int aux= cantdBloques - contarDigitos(indiceBloque);
                        for(int i=0; i<aux; i++){
                            fprintf(archivo,"-");
                        }
                        fprintf(archivo,"#0#");//indicador si esta ocupado o no
                        for(int i=0; i<cantdBloques; i++){ //bloque siguiente
                            fprintf(archivo,"-");
                        }
                        fprintf(archivo, "%i\n",espacioBloque-tamcabeceraBloque);
                        indiceBloque++;
                    }
                }
            }
        }
        fclose(archivo);
    }
}

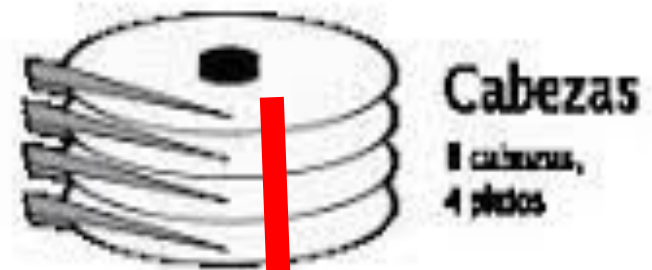
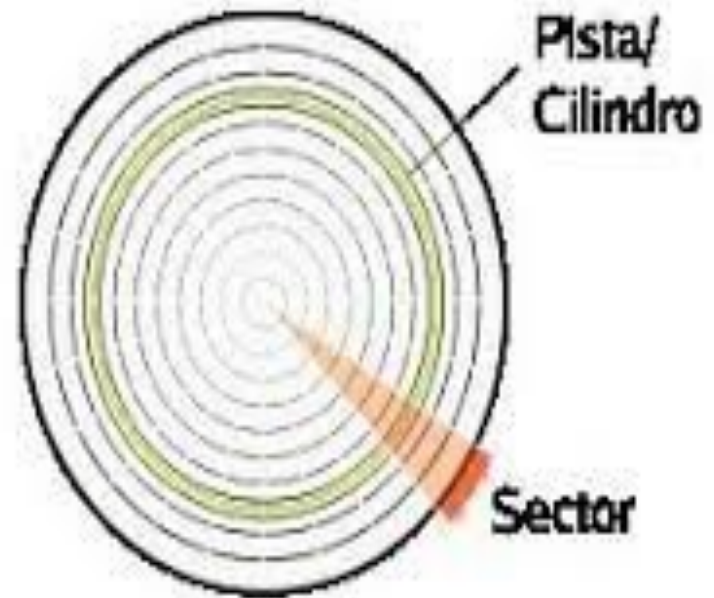
```

```

if(p==0 && sup==0){
    fprintf(archivo,"%i", indiceBloque); //indice bloque
    int aux= cantdBloques - contarDigitos(indiceBloque);
    for(int i=0; i<aux; i++){
        fprintf(archivo,"-");
    }
    fprintf(archivo,"#0#");//indicador si esta ocupado o no
    for(int i=0; i<cantdBloques; i++){ //bloque siguiente
        fprintf(archivo,"-");
    }
    fprintf(archivo, "##%i\n",espacioBloque-tamcabeceraBloque);
    indiceBloque++;
}

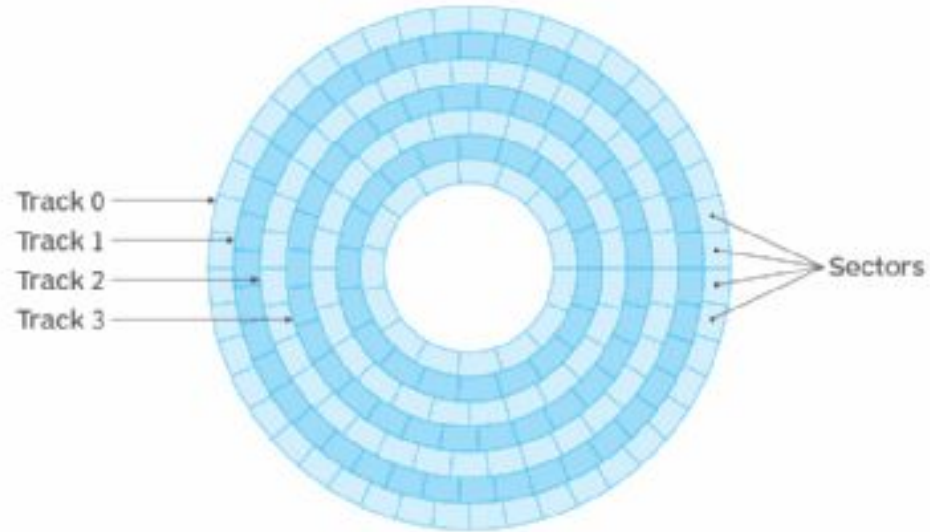
```

- Menos seek time (brazo se mueve menos)
- Se realiza cambio de cabezal lo cual suele ser más rápido ya que es electrónico



LBA

How logical block addressing works



diccionario
titanic#1
housing#2
(esquema)#(idBloque)

```
void MicroControlador::ObtenerRuta(int LBA) {  
    ruta[0] = LBA / (2*disco->pistas*disco->sectores); // Plato ..sectoresxplato  
    LBA = LBA%(2*disco->pistas*disco->sectores);  
    ruta[1] = LBA / (disco->pistas*disco->sectores); // Superficie ..sectoresxsuperficie  
    LBA = LBA%(disco->pistas*disco->sectores);  
    ruta[2] = LBA/disco->sectores; // Pista  
    ruta[3] = LBA % disco->sectores; // Sector  
}
```

LBA to CHS translation

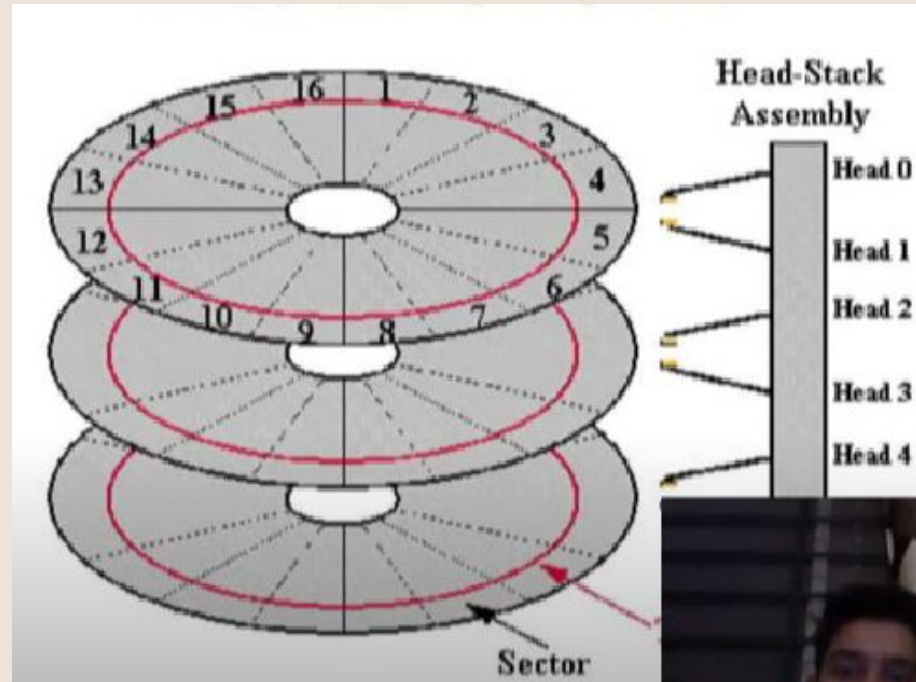
• Conversely LBA address can be translated into CHS address

$$\text{cylinder} = \text{LBA} / (\text{heads_per_cylinder} * \text{sectors_per_track})$$

$$\text{temp} = \text{LBA} \% (\text{heads_per_cylinder} * \text{sectors_per_track})$$

$$\text{head} = \text{temp} / \text{sectors_per_track}$$

$$\text{sector} = \text{temp} \% \text{sectors_per_track}$$



[illegible]

Metadata

- Cabecera bloque
- Datos del disco
- Bitmap de bloques

Campo	Descripción
idBloque	Identificador del bloque
Indicador	Si el bloque está vacío o lleno
Bloque siguiente	Enlace al siguiente bloque (ID)
Capacidad disponible	Espacio libre en el bloque

platos-pistas-sectoresxpista-tamañoSector-sectoresxbloque-espacioOcupado

10000000000000000000000000000000

Disco default:

- Platos: 4
- Pistas: 20
- Sectores: 8
- Sectores x bloque: 8
(proporcional a platos)
- Capacidad: 512 bytes

```
Capacidad del disco: 652800 Bytes
Capacidad ocupada: 2560Bytes
Capacidad del bloque: 4096 Bytes
Numero de bloques por pista: 1
Numero de bloques por plato: 40
Cantidad de pistas: 20
Cantidad de platos: 4
Cantidad de sectores: 8
```

```
+-- Sector7.txt
+-- Pista8/
+-- Sector0.txt
+-- Sector1.txt
+-- Sector2.txt
+-- Sector3.txt
+-- Sector4.txt
+-- Sector5.txt
+-- Sector6.txt
+-- Sector7.txt
+-- Pista9/
+-- Sector0.txt
+-- Sector1.txt
+-- Sector2.txt
+-- Sector3.txt
+-- Sector4.txt
+-- Sector5.txt
+-- Sector6.txt
+-- Sector7.txt
```

Referencias:

- <http://www.osdever.net/tutorials/view/lba-to-chs>
- Libro Database System Concepts
- Libro Database Systems The Complete Book
- ChatGPT