



Departamento de Informática
Universidad Técnica Federico Santa María



Unidad IV

Diseño Físico de Bases de Datos Relacionales

INF-239, ILI-239 Bases de Datos

Profesora Cecilia Reyes Covarrubias – Casa Central

Diapositivas realizadas con la colaboración Prof. J.Luis Martí – Campus San Joaquín



TEMARIO UNIDAD IV

4.1 Contexto Diseño Físico

4.2 Organización de Archivos

4.3 Índices

4.4 Método para el Diseño Físico de una BDR



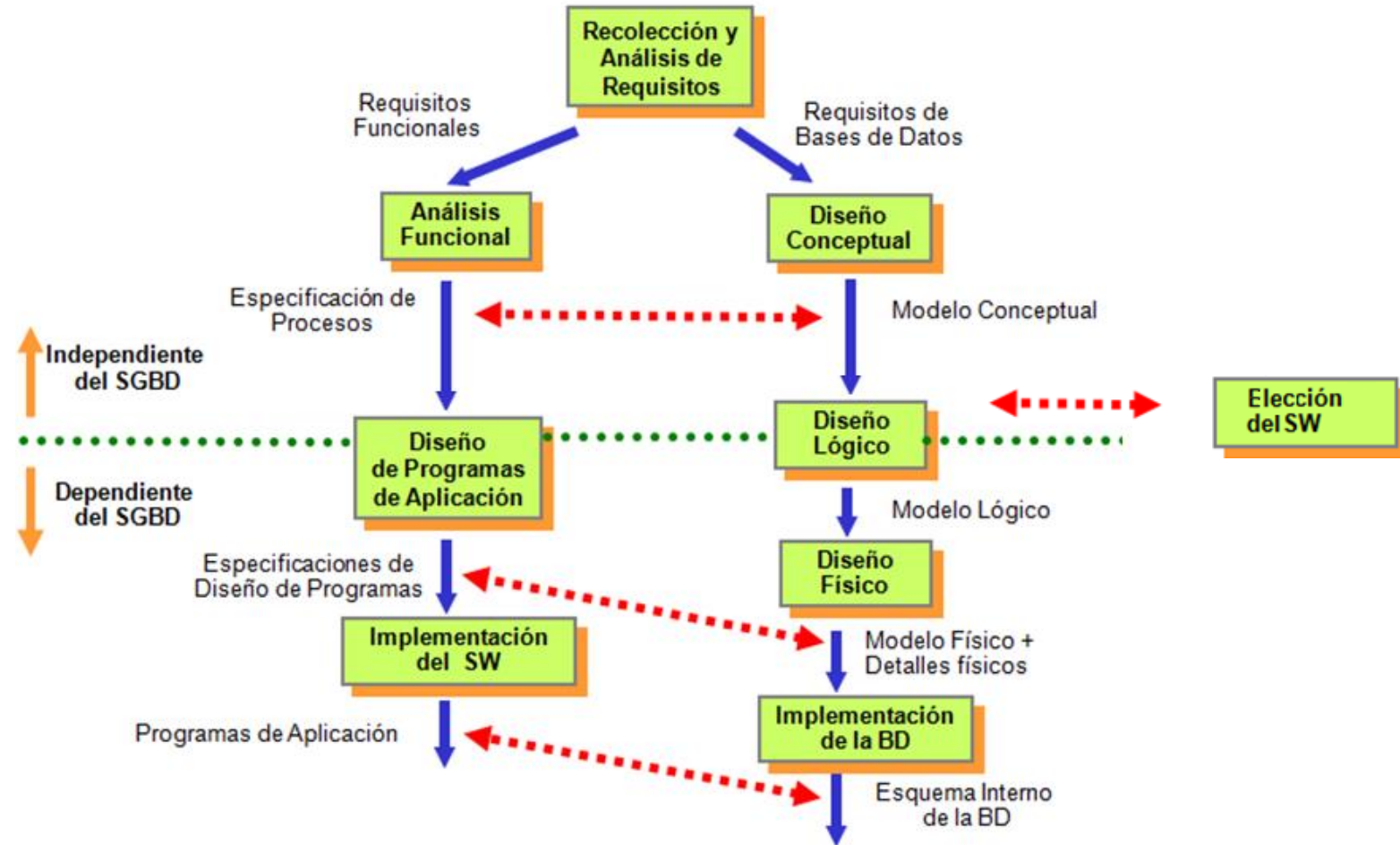
4.1 CONTEXTO DISEÑO FISICO



Departamento de Informática
Universidad Técnica Federico Santa María

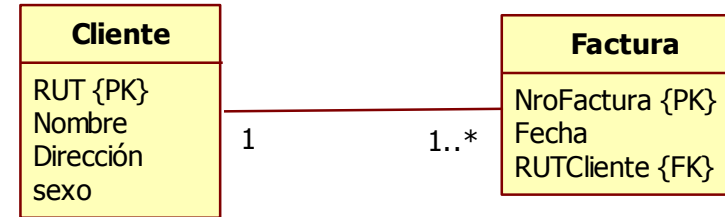


ETAPAS PARA DISEÑO DE BD



ETAPAS PARA DISEÑO DE BD

Modelo Lógico (relacional)



```
create table cliente
( RUT      char(9)  primary key,
  Nombre   varchar(35) not null,
  Direccion varchar(50)
  sexo     char);
```

```
create table factura
( NroFactura  number(7)  primary key,
  Fecha       date,
  RUTCliente  char(9)    foreign key
               references cliente(RUT));
```

Modelo Físico (relacional)

DISEÑO FISICO

- Esta etapa busca elegir las estructuras de almacenamiento específicas y caminos de acceso para los archivos (o tablas), con el objetivo de obtener un buen rendimiento de las aplicaciones que hacen uso de las distintas bases de datos.

DISEÑO FISICO

ASPECTOS A CONSIDERAR

- **Rendimiento** (performance) **del Sistema:** número promedio de transacciones que pueden ser procesadas por minuto por la base de datos.
- **Tiempo de Respuesta:** tiempo transcurrido entre la entrada de la transacción a la base de datos y la llegada de la respuesta.
- **Utilización del Espacio en Disco:** es la cantidad de memoria que los archivos y sus índices ocupan en los discos del sistema.

DISEÑO FISICO

ASPECTOS A CONSIDERAR

- **Análisis sobre:**
 - Consultas y transacciones.
 - Frecuencia esperada de la invocación de la consulta y transacciones.
 - Posibles restricciones de tiempo de las consultas y transacciones.
 - Frecuencias esperadas de las operaciones de actualización.
- El DBMS (SABD) es relevante en el diseño físico, cada DBMS permite una variedad de organizaciones de archivos y métodos de accesos.
 - Organizaciones de archivos: secuencial, *hashing*, organizados como árbol, ...
 - Índices: dinámicos, *bitmap*, ...

4.2 ORGANIZACIONES DE ARCHIVOS



Departamento de Informática
Universidad Técnica Federico Santa María



ARCHIVOS SECUENCIALES

Nombre	RUT	Fecha Nac.	Puesto	Sueldo	Sexo
Bloque 1					
<u>Wright, Ed</u>	6.576.354-7	22-3-45	Jefe Producción	550.000	M
Delaware, Diane
Clinton, Marc
Nixon, John
Bloque 2					
Adams, John
Wood, Robin
Chapman, Mark
Williams, Jan
Bloque 3					
<u>Roog, Ed</u>
<u>Alfred, Deborah</u>
<u>Murphy, Olivia</u>
Harris, Bob
Bloque 4					
Majors, Troy
Fawcett, Keith
<u>Fairchild, Farrah</u>
Stevens, Rob

SECUENCIAL DESORDENADO

Nombre	RUT	Fecha Nac.	Puesto	Sueldo	Sexo
Bloque 1					
<u>Aaron, Ed</u>	6.576.354-7	22-3-45	Jefe Producción	550.000	M
Abbott, Diane
...
<u>Acosta, Marc</u>
Bloque 2					
Adams, John
Adams, Robin
...
Akers, Jan
...					
Bloque N-1					
Wong, James
Wood, Donald
...
Woods, Manny
Bloque N					
<u>Wright, Pam</u>
Wyatt, Charles
...
<u>Zimmer, Byron</u>

SECUENCIAL ORDENADO

ARCHIVOS SECUENCIALES

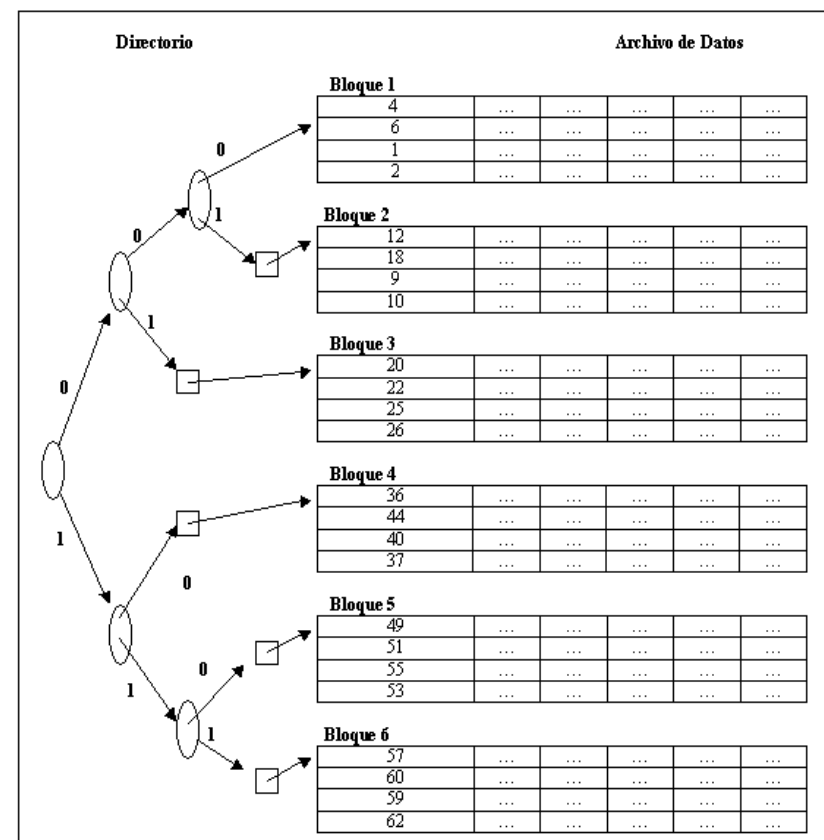
- Archivos **secuenciales recomendables** en aplicaciones con:
 - alta tasa de volatilidad de inserción (escrituras al final del archivo)
 - alta tasa de actividad
 - particularmente ordenados, en consultas de rango

```
create table cliente
( RUT      char(9)  primary key,
  Nombre   varchar(35) not null,
  Direccion varchar(50)
  sexo     char);
```

ARCHIVOS HASHING (directos)

	Nombre del Tema	Intérprete	#Estreno
0	Just Because	Jane's Adiction	880
	Show me How to Live	AudioSlave	450
1	White Flag	Dido	901
2	God puts a Smile upon your Face	Coldplay	342
	Sálvame la vida	Lucybell	1502
3	Frantic	Metallica	593
	Re-Offender	Travis	1473
4			
5	Hollywood	Madonna	125
6			
7	Go to Sleep	RadioHead	367
	Here we Kum	Molotov	287
8	Crazy in Love	Bevonce & Jay-Z	258
	San Miguel	Los Prisioneros	468
9			

ESTATICO



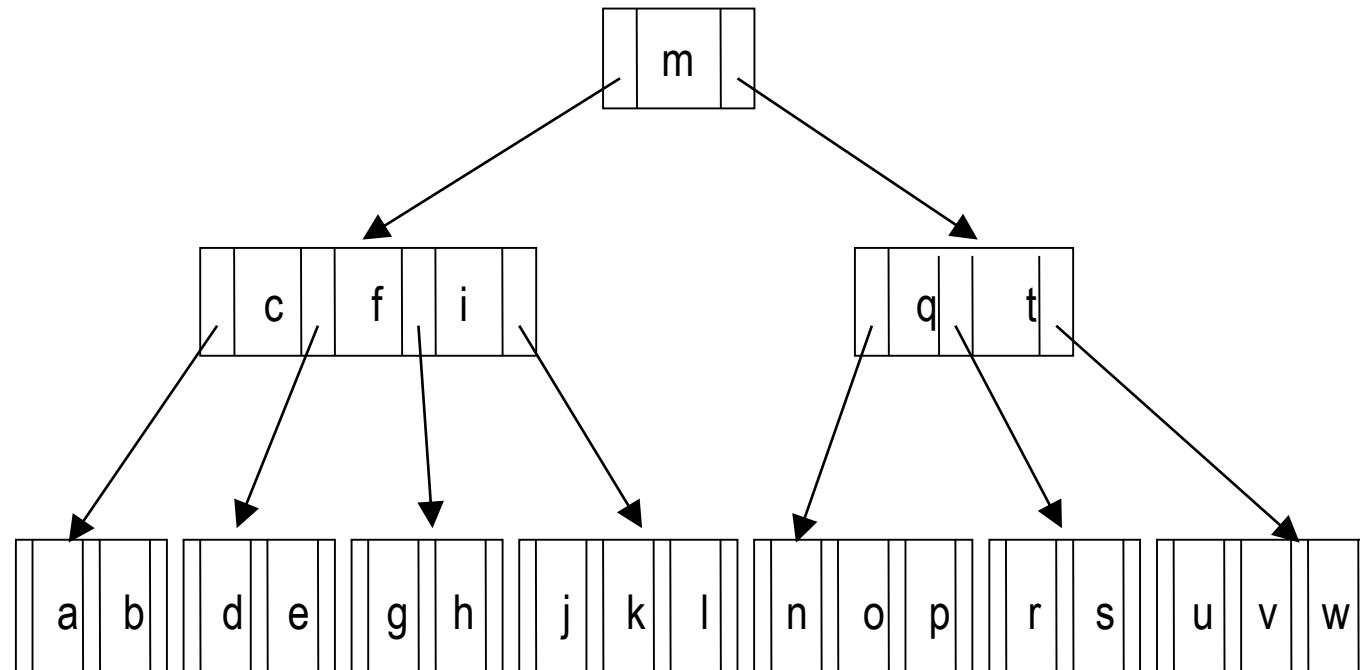
CON EXPANSIÓN DINÁMICA

ARCHIVOS HASHING (directos)

- Archivos hashing recomendables en aplicaciones con:
 - alta tasa de volatilidad, en archivos con expansión dinámica
 - baja tasa de actividad
 - no sirve, normalmente, para salidas ordenadas

```
create cluster cliente (codigo number)
( RUT      char(9)  primary key,
  Nombre   varchar(35) not null,
  Direccion varchar(50),
  sexo     char)
size 512 single table hashkeys 500;
```

ARCHIVOS ORGANIZADOS COMO ARBOLES

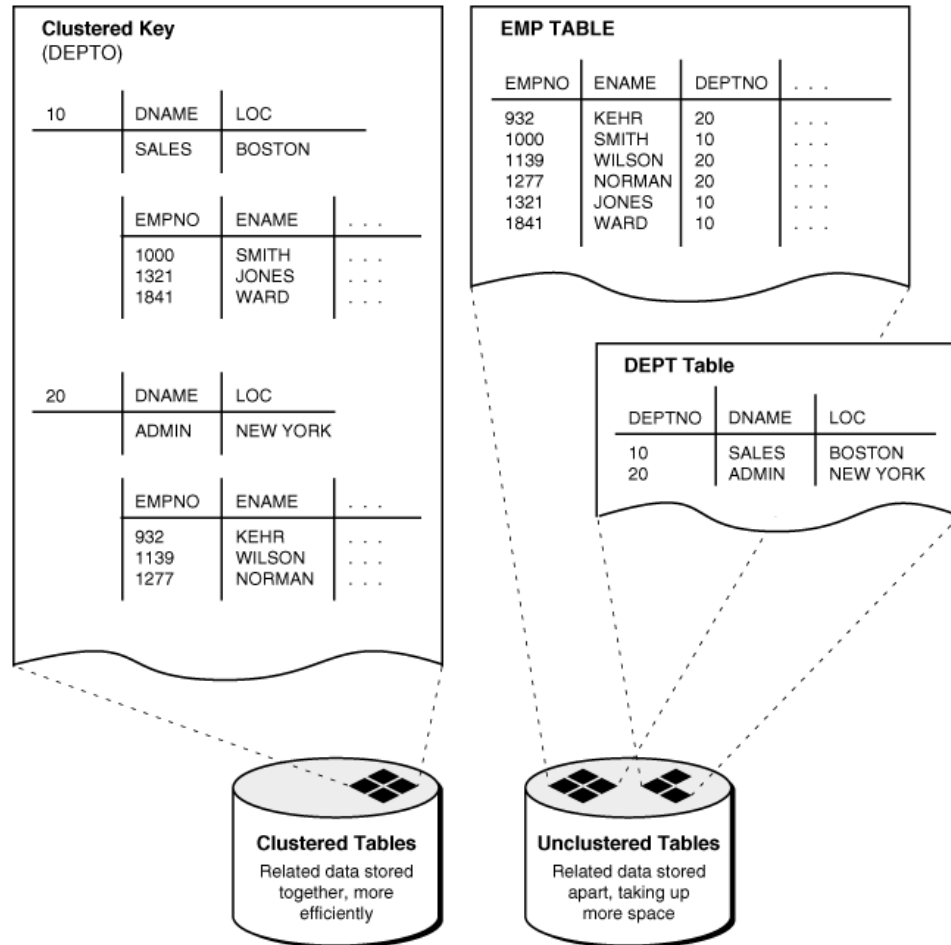


ARCHIVOS ORGANIZADOS COMO ARBOLES

- Archivos organizados como árboles recomendables en aplicaciones con:
 - alta volatilidad
 - búsquedas directas

```
create table cliente
( RUT      char(9)  primary key,
  Nombre   varchar(35) not null,
  Direccion varchar(50),
  sexo     char)
organization index;
```

CLUSTERS



Clusters, recomendados cuando:

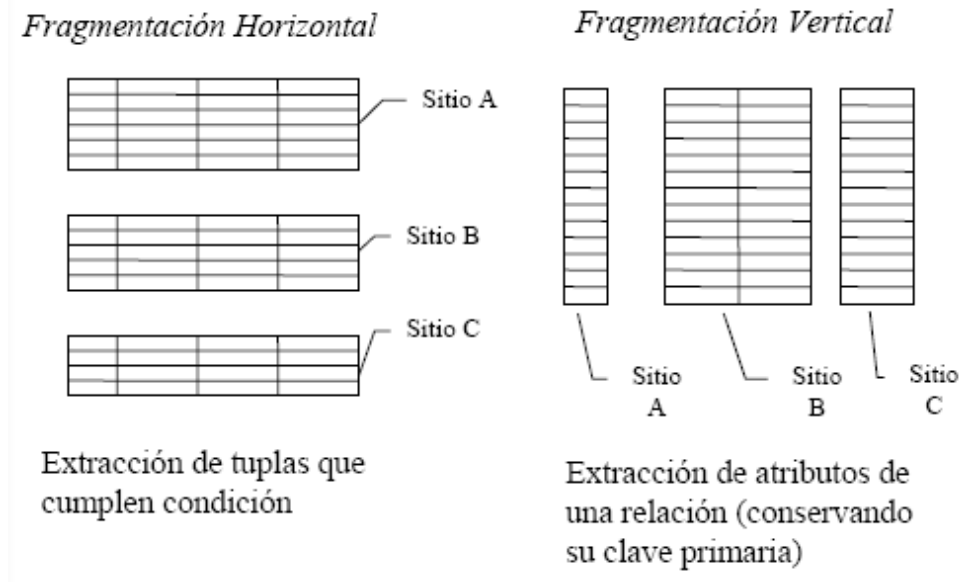
- tablas que son usadas principalmente de lectura
- registros de las tablas agrupadas son frecuentemente utilizados al mismo tiempo

```
create table empleado
( RUT      char(9)  primary key,
  Nombre   varchar(35) not null,
  ...
  nroDepto number(3) references depto
cluster emp_dept (nroDepto);
```

```
create table departamento
( nroDepto number(3) primary key,
  ...
cluster emp_dept (nroDepto);
```

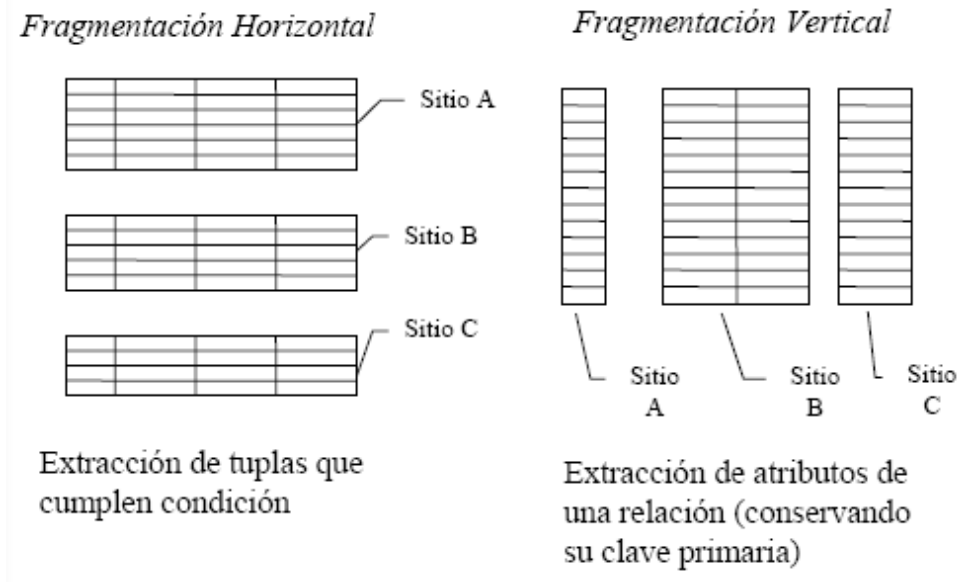

ARCHIVOS PARTICIONADOS

- Reducen el tiempo necesario para recuperar los datos de un archivo, repartiéndolos en partes de menor tamaño (y posiblemente en distintos discos).



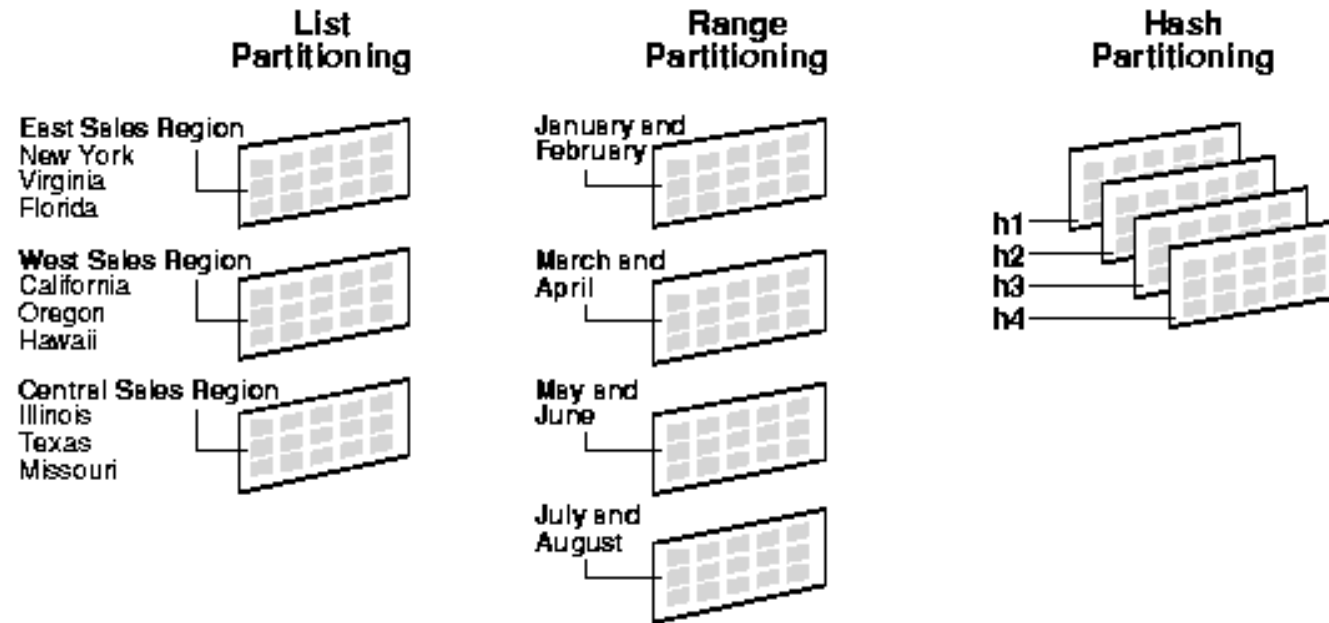
ARCHIVOS PARTICIONADOS

- Reducen el tiempo necesario para recuperar los datos de un archivo, repartiéndolos en partes de menor tamaño (y posiblemente en distintos discos).



ARCHIVOS PARTICIONADOS

HORIZONTALMENTE



ARCHIVOS PARTICIONADOS

HORIZONTALMENTE POR LISTA DE VALORES

```
create table ventas_por_listas  
( idVendedor      number(5),  
  nombreVendedor  varchar(30),  
  cantidadVendida number(10),  
  sucursal        varchar(20),  
  fechaVenta      date  
)
```

partition by list(sucursal)

```
(  
  partition ventas_norte_grande VALUES('Arica', 'Iquique'),  
  partition ventas_norte_grande VALUES ('Copiapo', ...),  
  partition ventas_centro VALUES(...),  
  partition ventas_sur VALUES(DEFAULT)  
)
```

[Arica, Iquique
Antofagasta]

[Copiapó,
La Serena]

...

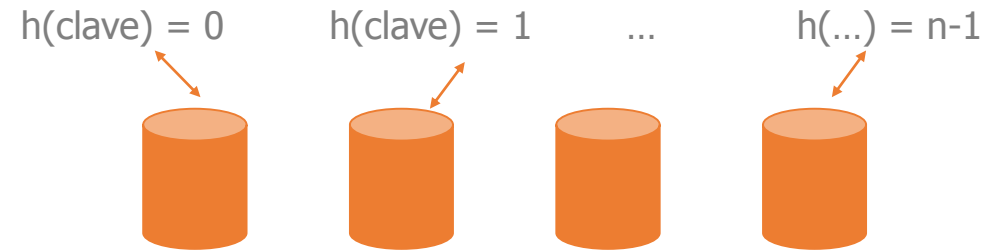
[Coyhaique,
Punta Arenas]



ARCHIVOS PARTICIONADOS

HORIZONTALMENTE POR CLAVES HASHING

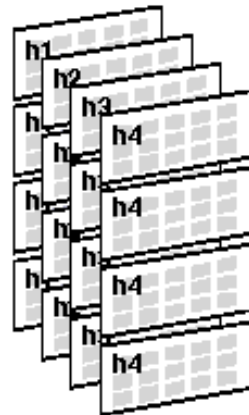
```
create table ventas_por_hashing  
( idVendedor      number(5),  
  nombreVendedor  varchar(30),  
  cantidadVendida number(10),  
  fechaVenta      date  
)  
partition by hash(idVendedor)  
partitions 4  
store in (datos1, datos2, datos3, datos4);  
);
```



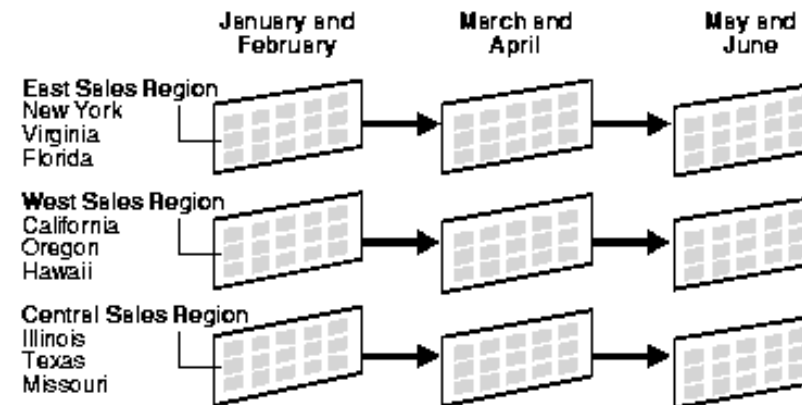
ARCHIVOS PARTICIONADOS

VIA COMBINACION TECNICAS ANTERIORES

Composite Partitioning
Range-Hash



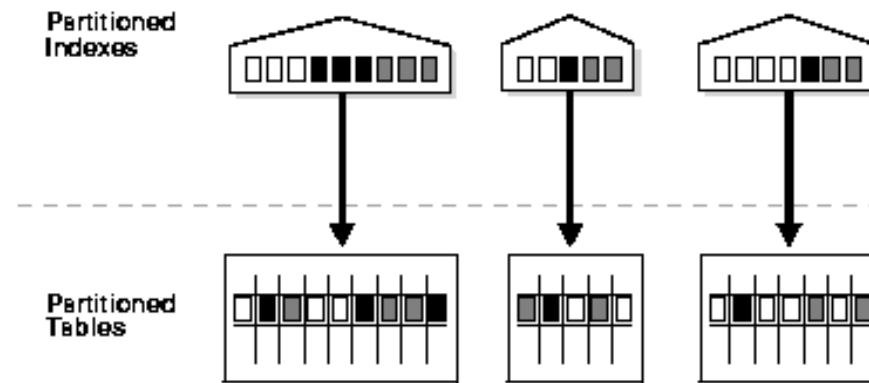
Composite Partitioning
Range - List



ARCHIVOS PARTICIONADOS

INDICES

- Archivos Particionados: permiten la posibilidad de tener índices particionados.



4.3 INDICES



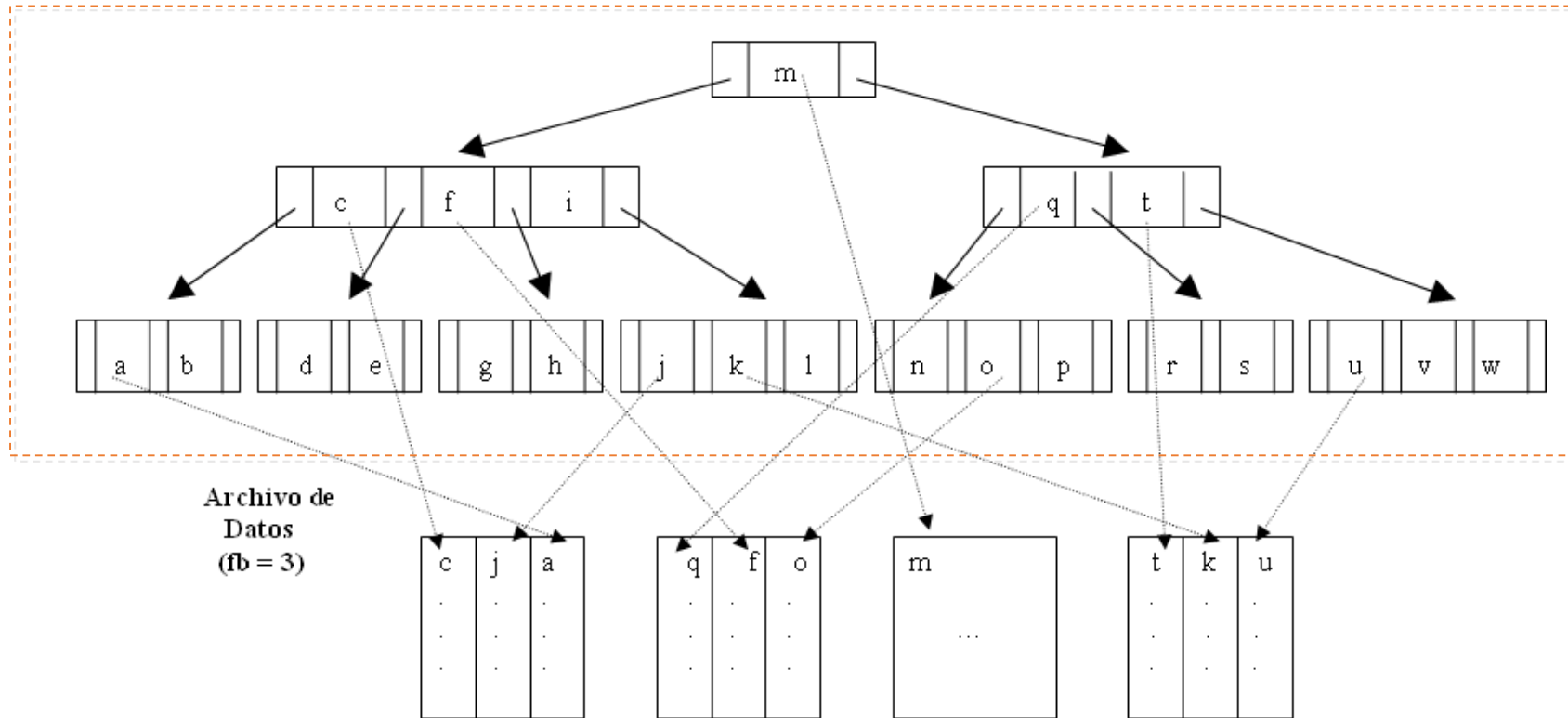
Departamento de Informática
Universidad Técnica Federico Santa María



INDICES

- Según el atributo sobre el cual se construye se denomina:
 - Índice Primario (Principal): basado en el atributo sobre el cual se organiza el archivo de datos, siendo también clave única.
 - Índice de Grupos: ídem que el anterior, pero siendo clave no único.
 - Índice Secundario: especificado sobre un campo que no es usado para organizar un archivo.
- Según su estructura:
 - Índice Dinámico
 - Índice *Bitmap*

INDICES DINAMICOS



Índices Dinámicos:

- atributos únicos
- amplio rango de valores

```
create table cliente  
( RUT      char(9)  primary key,  
  ... );
```

```
create index RC on cliente(RUT);
```

INDICES BITMAP

Referencias del Índice

1	0011
2	0011
3	1101
4	1000
5	1100
6	0110

Archivo

Página 1					
...	4
...	5
...	3
Página 2					
...	6
...	5
...	3
Página 3					
...	2
...	1
...	6
Página 4					
...	3
...	1
...	2

Índices *Bitmap*:

- atributos no únicos
- rango reducido de valores

```
create table cliente
( RUT      char(9)  primary key,
  ...
  sexo     char);
```

```
create bitmap index RC
on cliente(sexo);
```

INDICES BITMAP

CUSTOMER #	MARITAL_ STATUS	REGION	GENDER	INCOME_ LEVEL	'EAST'	'CENTRAL'	'WEST'
101	single	east	male	bracket_1	1	0	0
102	married	central	female	bracket_4	0	1	0
103	married	west	female	bracket_2	0	0	1
104	divorced	west	male	bracket_4	0	0	1
105	single	central	female	bracket_2	0	1	0
106	married	central	female	bracket_3	0	1	0

SELECT COUNT(*) FROM CUSTOMER
 WHERE MARITAL_ STATUS = 'married' AND REGION IN ('central','west');

status = 'married'		region = 'central'		region = 'west'						
0		0		0		0		0		0
1		1		0		1		1		1
1	AND	0	OR	1	=	1	AND	1	=	1
0		0		1		0		1		0
0		1		0		0		1		0
1		1		0		1		1		1

INDICES BITMAP

CUSTOMER #	MARITAL_ STATUS	REGION	GENDER	INCOME_ LEVEL	'EAST'	'CENTRAL'	'WEST'
101	single	east	male	bracket_1	1	0	0
102	married	central	female	bracket_4	0	1	0
103	married	west	female	bracket_2	0	0	1
104	divorced	west	male	bracket_4	0	0	1
105	single	central	female	bracket_2	0	1	0
106	married	central	female	bracket_3	0	1	0

SELECT COUNT(*) FROM CUSTOMER
 WHERE MARITAL_ STATUS = 'married' AND REGION IN ('central','west');

status = 'married'		region = 'central'		region = 'west'					
0		0		0		0		0	
1		1		0		1		1	
1		0		1		1		1	
0	AND	0	OR	1	=	0	AND	1	=
0		1		0		0		1	
1		1		0		1		1	

4.4 METODO PARA EL DISEÑO FISICO DE BDR



Departamento de Informática
Universidad Técnica Federico Santa María



METODO

- Escoger un atributo que sea frecuentemente usado para recuperar los registros en orden, o para realizar operaciones (Join), como el campo de ordenamiento del archivo:
 - Crear un índice primario o de grupos, según corresponda.
- En caso de que ningún atributo cumpla con las condiciones anteriores, mantener el archivo desordenado.
- Por cada atributo (que no sea el campo de ordenamiento), que es usado en operaciones de búsqueda o Join, especificar un índice que sirva como índice secundario al archivo de datos.
- Si el archivo va a ser actualizado frecuente-mente, con operaciones de inserción y eliminación de registros (alta volatilidad), tratar de reducir el número de índices del archivo de datos.

METODO

- Si un atributo es usado frecuentemente para operaciones de selección y Join, pero no para recuperar los registros en orden, un archivo tipo *hashing* puede ser usado.
 - Estático puede ser usado para archivos cuyo tamaño no varíe mucho.
 - Con Expansión Dinámica es requerido cuando el archivo sufrirá variaciones importantes en su volumen.
- Si el archivo tiene pocos registros, y éstos son de tamaño pequeño, podría considerarse una organización de tipo árbol.