Bases de Datos y Bases de Datos Avanzadas.

modelamiento de Bases de datos.

→ **entidad**: cualquier sujeto medible y comprensible. Contienen atributos.

→ **relacion**: asociacion entre entidades

→ **cardinalidad**: nivel de asociacion de la relación

- 1-1 ; 1-n ; n-1 ; n-m.

→ **dimension de medida**:

- bit a byte.
  - 1 byte = 8 bits
  - 1 Kbyte = 1024 bytes
  - 1 Mbyte = 1024 Kbytes
  - 1 Gbyte = 1024 Mbytes
  - 1 Tbyte = 1024 Gbytes

→ **llaves**:

atributos que identifican una unica ocurrencia de una entidad o relación

- super llaves → llaves candidatas; llaves primarias (identifican las tuplas en una tabla de forma unica).
- llaves foraneas → dependen de otra tabla.
  - clave que tiene valores que coinciden con los valores de clave primaria de otra relacion.

→ **Tipos de datos**:

tipo dato	tamaño almacenamiento	rango
• int	4 bytes	-2.147.483648 - 2.147.483.647
• VARCHAR	n + 1 bytes	0-255 caracteres
• year	1 byte	1901 - 2165 (anio)
• date	3 bytes	1/01/1001 — 31/12/9999
• DATETIME	8 bytes	1/01/1001 a las 0:0:0 — 31/12/9999 a las 23:59:59.
• TEXT	longitud + 2 bytes	0-255 caracteres
• TIMESTAMP	4 bytes.	1/01/1970 - 2037.

→ **Operadores unitarios**

- seleccion → (llamar a una tabla)
- proyeccion → (sacar atributos)
- renombramiento → (renombrar atributos y entidades).

→ **operadores binarios**:

- Resta, producto, asociacion, division, union, interseccion.

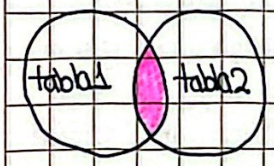
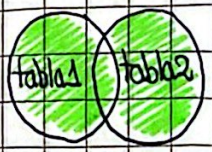
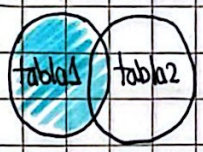
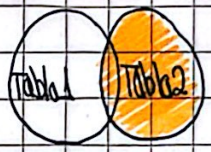
→ **estructura de una consulta JOIN**

```
SELECT <atributos>
FROM <tabla> (INNER|LEFT|RIGTH|FULL) JOIN
<tabla2> on <condition>
```

PROARTE®

→ estructura de una consulta

SELECT <atributo>
FROM <tabla>
WHERE <condition>

INNER JOIN	FULL JOIN	LEFT JOIN	RIGHT JOIN
			
TRAE RESULTADOS DE TABLAS QUE CUMPLEN CON LA COMPARACION ENTRE COLUMNAS	INFORMACION RESULTADOS DE TABLAS QUE CUMPLEN CON LAS COMPARACIÓN ENTRE COLUMNAS Y RESULTADOS DE TABLAS (DERECHA, IZQUIERDA)	COMPARACIÓN DE COLUMNAS Y TODOS LOS DATOS DE LA TABLA IZQUIERDA.	IGUAL QUE LEFT JOIN PERO CON LA TABLA DERECHA.

→ FUNCIONES INTEGRADAS

- MAX (<atributo>) : retorna el valor mayor.
- MIN (<atributo>) : retorna el valor menor.
- SUM (<atributo>) : suma de valores (entre ellos).
- AVG (<atributo>), promedio entre valores
- ORDER BY (<atributo>), ordena los resultados de una consulta segun el atributo indicado (lo hace de manera ascendente / DESC → descendente.
  ORDER BY <ATRIBUTO> DESC;
- COUNT (<atributo>), contador
- GROUP BY <(atributo>), agrupa los/ llos que tienen los mismos valores.
- LIMIT <n entero>; obtener el numero exacto de resultados
- BETWEEN (<valor1> AND <valor2>), rango inclusivo para un atributo).
- NOT BETWEEN (<valor1> AND <valor2>), negativo del BETWEEN
- IN / WHERE <atributo> IN (<valor1>, valor2>), permite buscar argumentos especificos de una lista.
- NOT IN, negativo del IN.

SELECT * FROM <tabla> where ....
SELECT <atributo> FROM <tabla>

BETWEEN / NOT BETWEEN

SELECT *
FROM <TABLA>
WHERE <atributo> BETWEEN / NOT BETWEEN
<valor1> AND <valor2>

IN / NOT IN

SELECT <atributo>
FROM <tabla>
WHERE <atributo> IN / NOT IN
(<valor1>, <valor2>).

PROARTE ©

→ NORMALIZACION

• conjunto de reglas obtenidas luego el paso del modelo entidad-relación las cuales busca:

→ Evitar la redundancia de datos
→ Evitar problemas de actualización de los datos en las tablas.
→ Proteger la integridad de los datos.
→ Agilizar las consultas o las actualizaciones o las inserciones o eliminaciones.

1FN o Primera Forma Normal

→ Todas las tablas tienen atributos conformados por un solo valor.

Ej:

nombre	rut	mascota
Hola1	1111-1	bruno, cachupin
Hola2	2222-2	colitor, maletin

↓

nombre	rut	mascota
hola1	1111-1	bruno
hola1	1111-1	cachupin
hola2	2222-2	colitor
hola2	2222-2	maletin

2FN o Segunda Forma Normal

→ Cumple el 1FN
→ Todos los atributos (que no sean parte de la clave candidata) entregan información de la clave completa.

Ej:

id mascota	nombre	rut	mascota	raza
113	hola1	1111-1	bruno	labrador
114	hola1	1111-1	cachupin	Puddle
225	hola2	2222-2	colitor	bulldog
227	hola2	2222-2	maletin	fox terrier

↓

id mascota	mascota	raza
113	bruno	labrador
114	cachupin	puddle
225	colitor	bulldog
227	maletin	fox terrier

id	rut
113	1111-1
119	1111-1
225	2222-2
227	2222-2

nombre	rut
hola1	1111-1
hola2	2222-2

PROARTE

3FN o Tercera Forma Normal

→ Cumple con 2FN

→ Todos los atributos dependen directamente de la llave primaria.

Ej.:

idmascota	mascota	raza
113	bruno	labrador
114	coliflor	fox temer
225	maletin	buldogg

idmascota	rut
113	1111-1
114	1111-1
225	2222-2

nombre	rut
Hola1	1111-1
Hola2	2222-2.

idmascota	mascota
113	bruno
114	coliflor
225	maletin.

mascota	raza
bruno	labrador
coliflor.	fox temer
maletin	buldogg

nombre	rut
hola1	1111-1
hola2	2222-2

idmascota	rut
113	1111-1
114	1111-1
225	2222-2

Bases de datos relacionales

→ Maneja información estructurada
→ Almacena la información en tablas (filas y columnas).
→ Establece relaciones entre tablas usando llaves.
→ Utiliza SQL como el lenguaje de administración y manejo de la bbda.
→ Orientado a sistemas transaccionales.
→ Garantiza las propiedades ACID (aislamiento, consistencia, durabilidad)

Ventajas:
- asegura la consistencia de los datos
- organiza los datos, elimina anomalías (duplicidad de datos, inconsistencias)
- dispone una gran variedad de herramientas que simplifican el manejo de la bbd

desventajas:
- orientada a trabajar en una sola maquina.
- escalabilidad limitada.
- esquema rigido de datos, difícil de modificar.
- rendimiento decae en la medida que la bbd crece.

Bases de datos no relacionales

→ maneja información no estructurada
→ almacena info usando estructuras de datos.
→ los datos no siguen un patron especifico.
→ almacena cualquier tipo de dato.
→ orientada al manejo de grandes volumenes de datos
→ datos no siguen un patron especifica.
→ sacrifica consistencia en pro de la eficiencia.
→ orientada a sistemas de gestion (OLAP).

Tipos:
- Key-values stores: diccionario no estructurado.
- Document Database: almacena documentos (JSON, BSON, XML)
- Graph Database: almacena datos en nodos.

ventajas:
- almacenamiento flexible de los datos
- permite manejar grandes volumenes de datos.
- posibilita la escalabilidad horizontal.
- simplifica la modificación del esquema de datos

desventajas:
- No existe lenguaje estandar de administra la base de datos
- NO asegura la consistencia de los datos
- Requiere una gran cantidad de almacenamiento debido a la redundancia de datos.

Vistas:

→ forma diferente de visualizar los datos físicos, se materializa mediante JOINS entre tablas.

→ visión lógica distinta de los datos físicos reales.

→ garantiza la independencia de datos y aporta confidencialidad, y sigue funcionando con normalidad si se hacen consultas.

→ se gasta lo que en los caracteres empleados para definirla. (solo)

ventajas:

- aporta confidencialidad a la BDD.
- si se estructura la BDD, alguna normalización, se puede mantener la estructura de las consultas en la BDD.

→ se crean mediante el comando view.

→ Los vistas (lógica) tiene una representación real (física) en una tabla.

VISTAS MATERIALIZADAS:

→ concentración de resultado de la consulta que genero la vista, es decir, la vista se crea y los resultados de esta vista quedan guardados en el disco, en la BDD.

→ se actualiza solo.

→ la vista materializada, no se actualiza se queda con los datos con la que se creo, es decir, no se puede borrar, ni agregar, etc.

PLAN DE EJECUCIÓN

→ se genera mediante el comando explain.

→ **componentes:**

1° • seq scan: secuencia de búsqueda por bloques.
- primer cost: costo del sistema para iniciar el proceso.
- costo: valor arbitrario.
- segundo cost: costo del sistema para terminar el proceso.
- rows: cantidad de filas que se recorre.
- width: cuantos bytes contiene cada fila.

2° Group Key = tabla de hash.

3° Nested loop = ciclo interno.

4° Index Only Scan = búsqueda logarítmica por que se utiliza un index.

→ solo búsqueda por índice con el Index Only Scan.

→ por cada una de las filas, se encuentra una, se demora un cost, encuentra solo 1 fila, por ende rows = 1 y width = 8.

5° Hash Aggregate = (cost = (cuanto cuesta), (total)).

Tipos de busqueda:

→ secuencial: busqueda por bloque.

→ bitmap index: sistema arma en memoria el recorrido necesario de lo que se pide sin ejecutarlo.

→ Index only: recorro solo el arbol, no necesito ir a buscar los datos

→ Index: busqueda por indice, navegar el B-tree y buscar los datos que necesito.

→ Bitmap heap: sistema guarda los datos en un heap.

Instrucciones de la BBD como funcionan.

SELECT

- Recibe un conjunto de condiciones que pueden ser de seleccion o de asociacion. Luego parte por las de seleccion con el fin de optimizar la consulta.
- Busca las filas que cumplan con la seleccion.
- Asocia esas filas con las otras tablas
- Esto implica una cantidad "N" de accesos (bloques secuencial como bloque del arbol del indice)
- Obtiene el resultado solicitado.

INSERT

caso 1: sin indice

- Trae a memoria el bloque ultimo de la tabla, si el bloque tiene espacio, agrega la fila a ese bloque.
- si el bloque no tiene espacio, genera un nuevo bloque y agrega la fila.
- Graba el bloque en el dispositivo de almacenamiento final de la tabla. +1 acceso.

caso 2: con indice

- se busca el nodo en donde se debe agregar el valor del atributo.
- una vez ubicado el nodo se trae a memoria, si hay espacio en ese nodo se agrega el nuevo valor.
- si no hay espacio, se realiza un block split (nodo se divide en 2, cada parte con la mitad de los valores por lo tanto se inserta el valor mas la direccion a memoria).

DELETE

- busca filas que cumplen con la condicion de seleccion.
- marcar en el bloque la eliminacion de la fila y luego grabar esto en el dispositivo de almacenamiento.
- si hay indice, se busca el nodo en donde se debe eliminar el valor y la direccion del atributo; elimina este valor y graba el nodo nuevamente en el dispositivo de almacenamiento.

UPDATE

- Buscar filas que cumplan con la condición de selección.
- Llevar a memoria el bloque que contiene la fila.
- La fila en cuestión se marca como eliminada y se crea otra fila con la actualización.
- Trae el último bloque en la tabla en cuestión, si hay espacio se agrega a la fila a ese bloque; si no hay espacio se genera un nuevo bloque para insertar la fila actualizada.
- Si existe un índice, se busca el nodo en cuestión, en el nodo se actualiza la dirección, el espacio será el mismo, así que no hay problema, luego se graba en el dispositivo de almacenamiento.

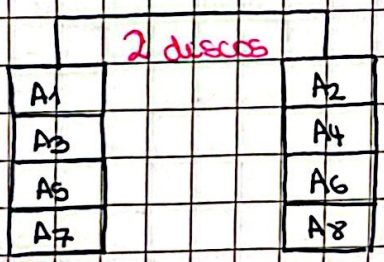
RAIDS: redundant array of independent disks

→ conjuntos de discos duros que se instalan en un rack, de manera que permite a la bbd ampliar su capacidad.

→ mejora el rendimiento de la base de datos.

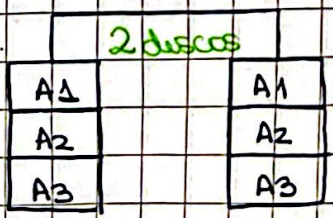
TIPOS DE RAID

- RAID 0: Data Striping, striped volume:



- volumen dividido, capacidad suma de discos.
- no tolera fallos, se guarda alternadamente.
- duplica la capacidad de almacenamiento, para lectura se tienen dos bloques o recuperar reduciendo el tiempo de acceso a la mitad.

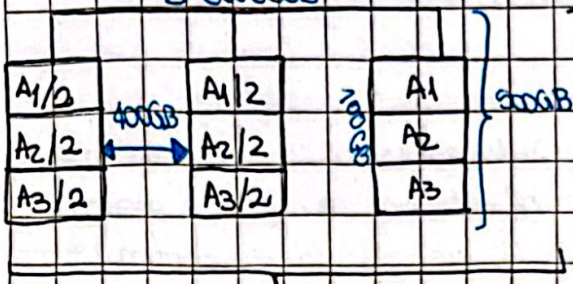
- RAID 1:



- disco espejo, no se pierde información y no se aumenta el espacio.
- se tienen datos redundantes.
- protección de datos, ya que se tienen los mismos datos.
- en lectura se pueden leer ambos discos, recuperando y reduciendo el tiempo.

PROARTE®

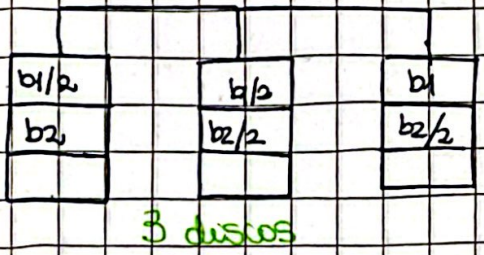
RAID 3: Disco extra de paridad "data striping".



- disco extra de paridad dedicado
- guarda la paridad de cada bloque
- si se cae uno de los 3 discos por medio del disco de paridad y del 2 restantes se recupera el perdido.

- capacidad: Σ de los discos - discos de paridad.
- soporta el fallo de un disco.
- duplica la capacidad de almacenamiento, es mas lento y recupera un disco en caso de perderse uno mediante el disco de chequeo y el restante.
- es mas lento que RAID 0 y RAID 1.

RAID 5: distribuido con paridad

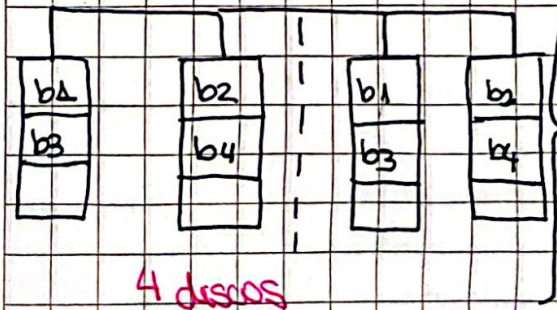


- paridad distribuida en todos los discos.
- capacidad suma de discos - disco de paridad.
- mejora lectura, penaliza escritura.
- soporta el fallo de un disco.
- funciona = que raid 3 pero el bloque de chequeo se puede ir moviendo en los discos.

RAID 6:

- funciona de la misma manera que RAID 5 pero el bloque de chequeo se va moviendo y son dos bloques de chequeo.
- se puede recuperar un disco, si se llegasen a perder dos.

RAID 10:



- se tiene un RAID 1 repetitivo, osea un espejo.
- se lee simultaneamente de 4 discos, por ende, el tiempo se divide en 4 osea menos.
- menos discos, menos espacio desperdiciado.
- se pierde menos espacio dependiendo el RAID.

PROARTE©

TABLESPACE

→ espacio en el disco en donde se almacena todos los objetos que crea la base de datos.

→ para el sistema operativo es solo un archivo que ocupa un determinado espacio extensible para la base de datos "estado" y postgress puede ver la estructura de este archivo.

→ limitado por el tamaño del disco.

→ no se debe llenar uno para crear otro, el sistema necesita espacio para procesos internos.

→ se crea mediante tablespace (comando).