IIC2413

AYUDANTÍA 12

Clase 10

BIENVENIDOS A

RECUPERACIÓN DE FALLAS

RECUPERACIÓN DE FALLAS

Después de una falla, la DB debe volver a un estado consistente

- Una transacción queda incompleta si el sistema falla antes del COMMIT.
- El DBMS usa el log para saber qué transacciones terminaron y cuales no.

En recuperación se decide:

- Undo: deshacer transacciones incompletas.
- Redo: rehacer transacciones confirmadas que podráin no haberse guardado en el disco.

Esto está reacionado a Atomicidad y Durabilidad

CHECKPOINTS

Un checkpoint permite no revisar todo el log desde el inicio.

- Marca un punto de referencia en el log.
- En Undo Logging, al iniciar un checkpoint se registran las transacciones activas.
- Para escribir <END CKPT> en Undo Logging, las transacciones activas al inicio del checkpoint deben terminar.
- En Redo Logging, antes de cerrar el checkpoint se guardan en disco los cambios confirmados hasta ese punto.

EJEMPLO

<START T1>
<T1, X, 100>
<START T2>
<T2, Y, 200>
<COMMIT T1>
--FALLA DEL SISTEMA--

Undo Logging

T1 hizo COMMIT → se conserva

T2 no hizo COMMIT → se deshace

En Undo Logging, el log guarda el valor antiguo

Acción de recuperación

UNDO T2:

Y:= 200

Redo Logging

T1 hizo COMMIT → debe quedar reflejada

T2 no hizo COMMIT → se ignora

En Redo Logging, el log guarda el valor nuevo

Acción de recuperación

UNDO T1:

X:= 100

EJERCICIOS

Undo logging. Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el *log file* que se muestra a continuación, en la tabla "Log Undo". Suponiendo que la política de *recovery* es la de *Undo Logging*, indique:

- Hasta qué parte del *log* debo leer.
- Qué variables deben deshacer sus cambios y cuál es el valor con el que quedarán.
- Qué variables (de las que aparecen en el *log*) no son cambiadas en el proceso.

Log Undo
<START T1>
<START T2>
<T1, a, 22>
<T2, b, 1>
<START T3>
<T2, b, 2>
<START T4>
<T4, c, 7>
<COMMIT T1>
<T3, d, 22>
<START CKPT (T2,T3,T4)>
<T3, a, 10>
<T2, b, 11>
<START T5>
<T5, d, 5>
<ABORT T4>
<T2, e, 32>
<COMMIT T2>
<T5, f, -3>
<COMMIT T5>

EJERCICIOS

Undo logging. Suponga que su sistema tuvo una falla. Al reiniciar el sistema, el sistema se encuentra con el *log file* que se muestra a continuación, en la tabla "Log Undo". Suponiendo que la política de *recovery* es la de *Undo Logging*, indique:

- Hasta qué parte del *log* debo leer.
  - Cómo hay un `START CKPT (T2,T3,T4)` sin su `END CKPT` respectivo, leemos desde la transacción más antigua entre `T2,T3,T4` (en este caso `T2`).
- Qué variables deben deshacer sus cambios y cuál es el valor con el que quedarán.
  - Transacciones `T1,T2,T4` y `T5` están finalizadas. Por lo tanto, solo hay que hacer el `UNDO` de `T3`. Las únicas variables que cambiarán en este proceso son `d` y `a`.
- Qué variables (de las que aparecen en el *log*) no son cambiadas en el proceso.
  - El resto.

Log Undo
<START T1>
<START T2>
<T1, a, 22>
<T2, b, 1>
<START T3>
<T2, b, 2>
<START T4>
<T4, c, 7>
<COMMIT T1>
<T3, d, 22>
<START CKPT (T2,T3,T4)>
<T3, a, 10>
<T2, b, 11>
<START T5>
<T5, d, 5>
<ABORT T4>
<T2, e, 32>
<COMMIT T2>
<T5, f, -3>
<COMMIT T5>

EJERCICIOS

Redo logging. Considerando el *schedule*:

Indique:

- Desde qué parte del *log* debo comenzar el proceso de *redo*.
- Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán.
- Qué variables (de las que aparecen en el *log*) no son cambiadas en el proceso.
- Si no hubiesemos encontrado la línea `<END CKPT>`, ¿desde qué parte del *log* debería comenzar el proceso de *redo*?.

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3>
<END T1>
<T3, a, 10>
<START CKPT (T3)>
<T3, d, 23>
<START T4>
<END T2>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

EJERCICIOS

Redo logging. Considerando el *schedule*:

Indique:

- Desde qué parte del *log* debo comenzar el proceso de *redo*.
  - Dado que existe un `END CKPT`, debo leer hasta el `<START T3>`, que es la transacción más antigua que se señala en el `START CKPT`.
- Qué variables deben rehacer sus cambios y cuál es el valor con el que quedarán.
  - Tenemos la certeza de que T1 y T2 están guardadas en disco, porque tenemos la presencia de un `END CKPT`. Fuera de esas transacciones, debemos rehacer todas las otras transacciones que están marcadas con `COMMIT` y no tienen un `END`. En este caso es sólo T3. Por lo que `<T3, a, 10>` nos indica que debemos rehacer **a** al valor 10 y `<T3, d, 23>` nos indica que d debe rehacer **d** a 23.
- Qué variables (de las que aparecen en el *log*) no son cambiadas en el proceso.
  - Continuando con el proceso anterior, T4 no está marcado con commit, por lo que **e** no se debe tocar. T4 debe ser marcada con `ABORT` en el proceso. **b** y **c** tampoco son cambiadas en el proceso. Lo cambios efectuados a la variable **a** debido a T1 tampoco deben ser realizados de nuevo.
- Si no hubiesemos encontrado la línea `<END CKPT>`, ¿desde qué parte del *log* debería comenzar el proceso de *redo*?.
  - Debemos encontrar sí o sí un `<END CKPT>`. Así que como no hay otro se debe leer el *log* entero.

Log Redo
<START T1>
<T1, a, 1>
<COMMIT T1>
<START T2>
<T2, b, 2>
<T2, c, 3>
<COMMIT T2>
<START T3>
<END T1>
<T3, a, 10>
<START CKPT (T3)>
<T3, d, 23>
<START T4>
<END T2>
<END CKPT>
<COMMIT T3>
<T4, e, 11>

Clase 11

BIENVENIDOS A

ÍNDICES

MODELO DE COSTOS E I/O

El costo de una consulta se mide por cuántas páginas se leen desde el disco.

- Lo más lento en una DB es trar datos desde el disco a la RAM.
- El modelo de costos cuenta accesos de lectura/escritura al disco.
- Los accesos a la RAM se consideran despreciables
- La lectura se hace por páginas, no por tuplas individuales
- Si una página almacena B tuplas, leer n tuplas cuesta aproximadamente:
  ceil(n / B) páginas

CÓMO SE GUARDA UNA TABLA

Una tabla se almacena en disco repartida en páginas.

- Las tuplas de una relación se agrupan en páginas.
- Aunque una consulta pida una sola tupla, puede ser neceasrio leer una página completa.
- Si no hay índice, una búsqueda normalmente debe revisar todas las páginas de la tabla.
- Ej: una tabla con 9 tuplas y 3 páginas tiene costo 3 para SELECT * FROM T.

Sin índice
Ejemplo:
SELECT * FROM T WHERE T.a=4;
Sin índice:
Costo = números de T = celi(|T|/ B)

“Sin índice, incluso una búsqueda exacta puede obligar a recorrer toda la tabla”

ÍNDICES Y HASH

Un índice permite encontrar datos sin recorrer toda la tabla. Un índice puede devolver:

- La tupla buscada directamente, o
- Un puntero a la página donde está la tupla

Un índice hash usa una función h(k) para mapear una llave a una posición.
Es muy eficiente para búsquedas exactas, por ejemplo:
SELECT * FROM T WHERE a 457;
Pero es malo para rangos, porque el hash no mantiene orden.

Ejemplo:
h(15) = 15 mod 7 = 1
h(37) = 37 mod 7 = 2
h(51) = 51 mod 7 = 2 → colisión
“Hash es ideal cuando se busca igualdad, pero no sirve bien si se quiere consultar rangos como menor que o entre dos valores”.

ÁRBOLES B+

Los árboles B+ son índices ordenados y eficientes para búsquedas exactas y de rango.

- Un B+ Tree es un índice basado en árbol
- Las páginas del disco representan nodos del árbol
- Las tuplas o punteros se encuentran en las hojas
- Tiene búsqueda logarítmica: costo aproximado O(h), donde h es la altura del árbol
- Funciona muy bien para consultas de rango

Ejemplo:
SELECT * FROM T WHERE c < 12;

Con B+ Tree se puede llegar al primer valor relevante y recorrer las hojas ordenadas.

“El B+ Tree es más general que hash, porque permite búsquedas exactas y también rangos”.

Clase 12

BIENVENIDOS A

ALGORITMOS

COSTO DE UN ALGORITMO

Operaciones en buffer (RAM) – costo 0

Leer / escribir al disco – mayor costo

ALGORITMOS EN LA BD

Implementan una interfaz de un iterador lineal:

- open()
- next()
- close()

Para una relación R:

- R.open() – se coloca antes de la 1° tupla de R.
- R.next() – devuelve la siguiente tupla o NULL.
- R.close() – cierra el iterador.

OPERADORES LINEALES

Selección: El algoritmo depende si la consulta es de igualdad (=) o de rangos (< , >). Además, depende si el atributo está indexado.

Proyección: El algoritmo es muy secillo. Necesariamentes hay que recorrer todo R. (project(t, atributos))

Joins: Operación muy costosa. Supondremos solamente restricciones de igualdad (R.a = R.b)

Nested Loop Join: Realiza join entre R y S, debe satisfacer un predicado p. (Block Nested Loop Join: aprovecha mejor el buffer)

→

```
open()
    R.open()

next()
    t:= R.next()
    while t != null do
        return project(t, atributos)
    return null

close()
    R.close()
```

```
next()
    while r != null do
        s:= S.next()
        if s == null then
            S.close()
            r:= R.next()
            S.open()
        else if (r, s) satisfacen p then
            return (r, s)
    return null
```

Costo I/O : costo(R) + tuplas(R) x costo(S)
Costo I/O : costo(R) + (pag(R)/bufer-2) x costo(S)

SORTING

Merge: Mezcla de secuencias ordenadas, forman una nueva mezcla ordenada.

Sigue la estrategia de “Dividir para conquistar”.

Merge Sort:

1. Divide la secuencia en dos sub-secuencias.
2. Utiliza llamadas recursivas.
3. combina las sub-secuencias ordenadas mediante Merge.

External Merge Sort: Funciona con frases.

- Run: secuancia de páginas que contiene un conjunto ordenado de tuplas.
- Pasadas = 1 + ceil(log_{M-1} (B/M))

Fase 0 : creamos los runs iniciales.
Fase 1 :

- traemos los runs a memoria.
- hacemos merge de cada par de runs.
- almacenamos el nuevo run a disco.

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$R(\text{cliente}, b)$:	80.000 tuplas, 40 /pág.
$S(b, c)$:	320.000 tuplas, 80 /pág.
$T(c, d)$:	640.000 tuplas, 80 /pág.

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(a) Calcule P_R, P_S y P_T.

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$$\begin{aligned} R(\text{cliente}, b): &\quad 80.000 \text{ tuplas, } 40 \text{ /pág.} \\ S(b, c): &\quad 320.000 \text{ tuplas, } 80 \text{ /pág.} \\ T(c, d): &\quad 640.000 \text{ tuplas, } 80 \text{ /pág.} \end{aligned}$$

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(a) Calcule P_R, P_S y P_T.

Solución:

$$\begin{aligned} P_R &= \frac{80.000}{40} = 2.000, \\ P_S &= \frac{320.000}{80} = 4.000, \\ P_T &= \frac{640.000}{80} = 8.000. \end{aligned}$$

Ahora, se quiere evaluar primero $R \bowtie S$ por b, y luego $(R \bowtie S) \bowtie T$ por c. Para esto se tienen los siguientes planes:

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$R(\text{cliente}, b)$:	80.000 tuplas, 40 /pág.
$S(b, c)$:	320.000 tuplas, 80 /pág.
$T(c, d)$:	640.000 tuplas, 80 /pág.

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(b) Plan A (todo BNLJ): primero $R \bowtie S$ con BNLJ (R externa), luego $(RS) \bowtie T$ con BNLJ (RS externa). Calcule el costo total.

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$$\begin{aligned} R(\text{cliente}, b): &\quad 80.000 \text{ tuplas, } 40 \text{ /pág.} \\ S(b, c): &\quad 320.000 \text{ tuplas, } 80 \text{ /pág.} \\ T(c, d): &\quad 640.000 \text{ tuplas, } 80 \text{ /pág.} \end{aligned}$$

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(b) Plan A (todo BNLJ): primero $R \bowtie S$ con BNLJ (R externa), luego $(RS) \bowtie T$ con BNLJ (RS externa). Calcule el costo total.

Solución: Recordatorio BNLJ:

$$\text{Costo}_{\text{BNLJ}} = P_{\text{ext}} + \left\lceil \frac{P_{\text{ext}}}{M-2} \right\rceil \cdot P_{\text{int}}, \quad M - 2 = 18.$$

1) $R \bowtie S$: $\lceil \frac{2.000}{18} \rceil = 112$. Costo $= 2.000 + 112 \cdot 4.000 = 450.000$.

2) $(RS) \bowtie T$: $P_{RS} = 4.000$, $\lceil \frac{4.000}{18} \rceil = 223$. Costo $= 4.000 + 223 \cdot 8.000 = 1.788.000$.

Total: $450.000 + 1.788.000 = 2.238.000$ I/Os.

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$R(\text{cliente}, b)$:	80.000 tuplas, 40 /pág.
$S(b, c)$:	320.000 tuplas, 80 /pág.
$T(c, d)$:	640.000 tuplas, 80 /pág.

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(c) Plan B (Sort-Merge en cadena): ordenar R y S por b y hacer *Merge Join* $R \bowtie S$, luego ordenar RS por c y ordenar T por c, y finalmente *Merge Join* $(RS) \bowtie T$. Calcule el costo total.

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$$\begin{aligned} R(\text{cliente}, b): &\quad 80.000 \text{ tuplas, } 40 \text{ /pág.} \\ S(b, c): &\quad 320.000 \text{ tuplas, } 80 \text{ /pág.} \\ T(c, d): &\quad 640.000 \text{ tuplas, } 80 \text{ /pág.} \end{aligned}$$

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(c) Plan B (Sort-Merge en cadena): ordenar R y S por b y hacer *Merge Join* $R \bowtie S$, luego ordenar RS por c y ordenar T por c, y finalmente *Merge Join* $(RS) \bowtie T$. Calcule el costo total.

Solución: Con $M = 20 \Rightarrow M - 1 = 19$. Usamos Pasadas $= 1 + \lceil \log_{19}(\frac{B}{20}) \rceil$, Costo sort $= 2B(\text{Pasadas} + 1)$.

Ordenar R: $B = 2.000$, $\frac{B}{M} = 100$, $\log_{19}(100) \approx 1.56 \Rightarrow$ Pasadas $= 3$. Costo $= 2 \cdot 2.000 \cdot 4 = 16.000$.

Ordenar S: $B = 4.000$, $\frac{B}{M} = 200$, $\log_{19}(200) \approx 1.80 \Rightarrow$ Pasadas $= 3$. Costo $= 32.000$.

Merge $R \bowtie S$: lectura $P_R + P_S = 6.000$.

Ordenar RS **por** c: $P_{RS} = 4.000 \Rightarrow$ costo $= 32.000$.

Ordenar T **por** c: $B = 8.000$, $\frac{B}{M} = 400$, $\log_{19}(400) \approx 2.03 \Rightarrow$ Pasadas $= 4$.

Costo $= 2 \cdot 8.000 \cdot 5 = 80.000$.

Merge $(RS) \bowtie T$: lectura $P_{RS} + P_T = 12.000$.

Sumas: Sorts $= 16.000 + 32.000 + 32.000 + 80.000 = 160.000$; Merges $= 6.000 + 12.000 = 18.000$.

$$160.000 + 18.000 = 178.000 \text{ I/Os.}$$

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$R(\text{cliente}, b)$:	80.000 tuplas, 40 /pág.
$S(b, c)$:	320.000 tuplas, 80 /pág.
$T(c, d)$:	640.000 tuplas, 80 /pág.

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(d) ¿Qué plan elegirías y por qué?

EJERCICIOS

Considere tres relaciones y un buffer de $M = 20$ páginas. Los tamaños son:

$R(\text{cliente}, b)$:	80.000 tuplas, 40 /pág.
$S(b, c)$:	320.000 tuplas, 80 /pág.
$T(c, d)$:	640.000 tuplas, 80 /pág.

No hay índices y las tablas no están ordenadas inicialmente. Suponga (caso base) que **cada tupla de** R matchea, en promedio, con **4** tuplas de S (estilo FK), por lo que $|R \bowtie S| \approx 4|R| = 320.000$ tuplas.

(d) ¿Qué plan elegirías y por qué?

Solución: Escoger **Plan B (Sort-Merge en cadena)**. Su costo **178.000** I/Os es *muy* inferior a **2.238.000** I/Os del Plan A. Ordenar para habilitar *merge* lineal es decisivo en este escenario.

Clase 13

BIENVENIDOS A

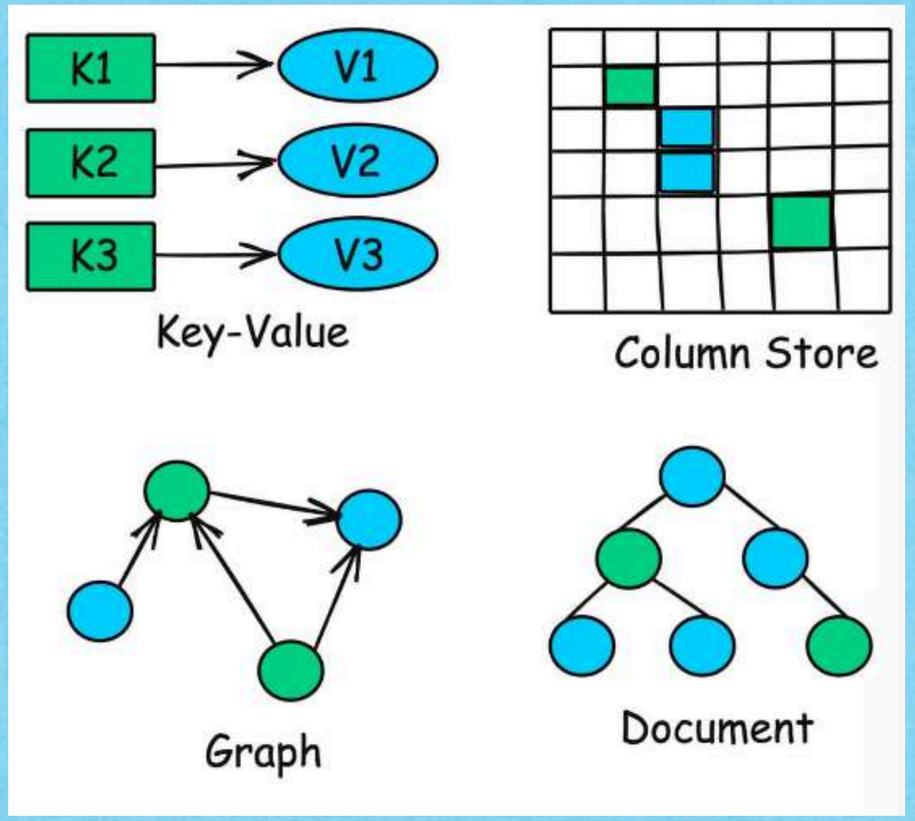
NO SQL

Tipo de base de datos

NO SQL

Es una amplia variedad de sistemas de gestión de bases de datos que no utilizan el modelo relacional tradicional

- Esquema flexible (o sin esquema)
- No pueden manejar grandes volumenes de datos.



BD KEY - VALUE

Son grandes tablas de hash persistentes.

Almacena información por medio de pares (key, values).

Operaciones:

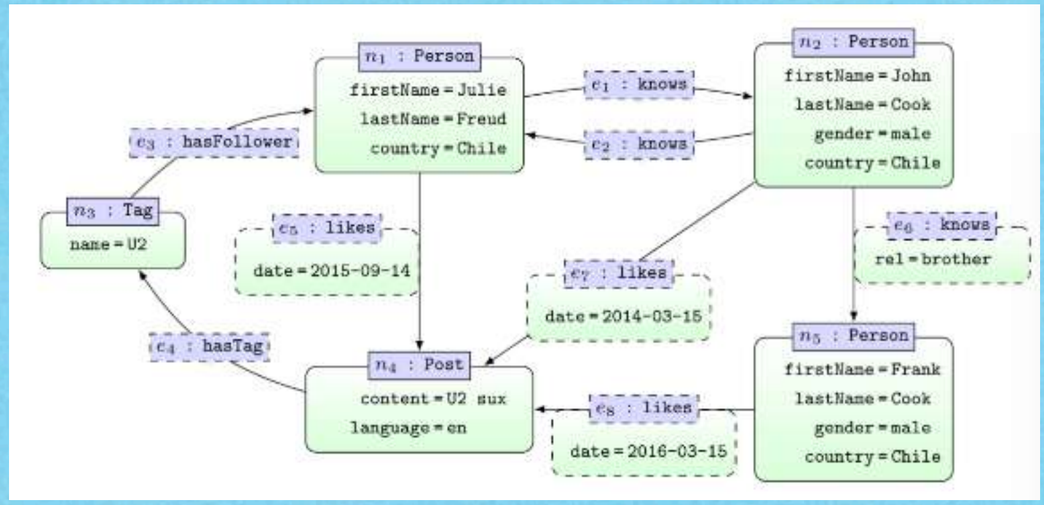
- put(key, value)
- get(key)
- delete(key)

Key	Value
Chile	Santiago
Inglaterra	Londres
Escocia	Edinburgo
Francia	Paris
Alemania	Berlin
...	...

BD DE GRAFOS

Guarda relaciones.

Almacena sus datos como grafos de propiedades.



BD ORIENTADAS A DOCUMENTOS

Almacenan sus datos en documentos JSON (MongoDG).

BD parecida a key – value:

- El valor es un documento.
- Pueden agrupar documentos en colecciones.
- Lenguaje de consultas mucho más poderosas.

Usuarios	
Key	JSON
1	
2	...
...	...

JSON

```
{
  "profesores": [
    {"nombre": "Juan", "apellido": "Reutter"},
    {"nombre": "Cristian", "apellido": "Riveros"},
    {"nombre": "Marcelo", "apellido": "Arenas"}
  ]
}
```

Es más flexible que SQL, y no necesariamente respeta un esquema.

MONGO BD

Es una base de datos NoSQL orientada a documentos que almacena en formato JSON

- Flexibilidad de esquema: almacena documentos con diferentes estructuras.
- Estabilidad Horizontal: distribuye datos en múltiples servidores para mejorar rendimiento y capacidad.
- Alto rendimiento: operaciones rapidas de lectura y escritura.
- Replicación y alta disponibilidad: asegura disponibilidad y recuperación ante fallos mediante réplicas.

Usuarios	
Key	JSON
60bfd90e002ce228636e506b	
60bfd90e002ce228636e5215	...
...	...

BÚSQUEDA Y RECUPERACIÓN DE DATOS

Ctrl + F

Para hacer una busqueda eficiente utilizamos índices invertidos.

Para cada palabra del universo de documentos, guardamos punteros que indican dónde están los documentos.

	Documento 1	Documento 2	Documento 3	...
Chile	1	0	1	
of	1	1	1	
town	0	0	1	
commune	0	1	0	
...				

TF (Frecuencia de Términos)

El puntaje es proporcional a la cantidad de veces que aparece la palabra en el documento.

$F_D(t)$ = Número de veces que aparece t en D

IDF (Frecuencia Inversa de los Documentos)

El puntaje es inversamente proporcional a la cantidad de documentos en los que aparece la palabra.

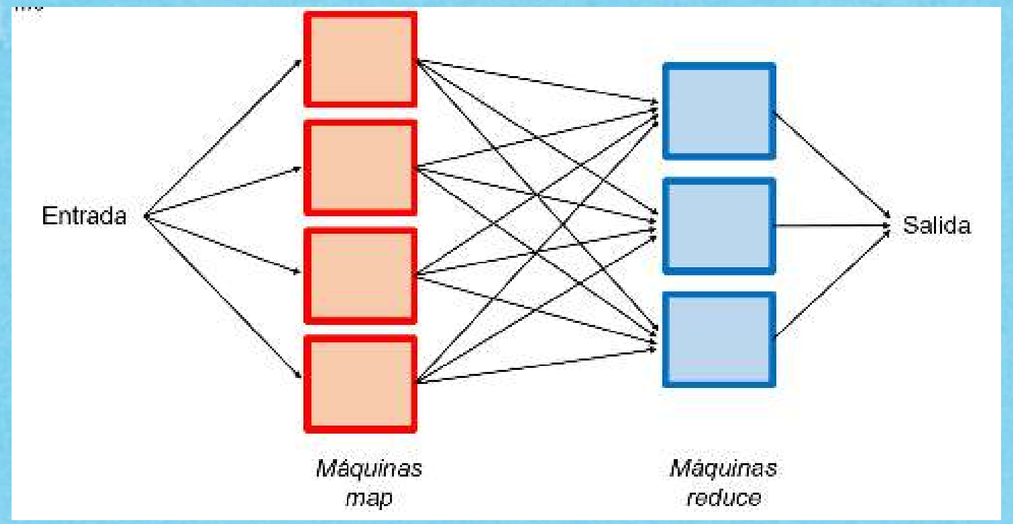
IDF(t) = log(número de documentos / número de documentos en los que aparece t)

TF-IDF = $F_D(t) \cdot IDF(t)$

MAP REDUCE

Algoritmo eficiente para computación distribuida (datos tan grandes que deben estar repartidos en muchos computadores, cada servidor no conoce lo que tiene el resto).

- Map: recibe datos y genera pares key – values.
- Reduce: recive pares con el midmo key y los agrega.
- Shuffle: transfiere los datos desde mappers a reducers.



EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** “El gato come pescado”
- **Documento 2:** “El perro come carne”
- **Documento 3:** “El gato y el perro son amigos”

a) Calcular el TF (Frecuencia de Término): para los términos gato y perro (0,1 pts)

EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** "El gato come pescado"
- **Documento 2:** "El perro come carne"
- **Documento 3:** "El gato y el perro son amigos"

a) Calcular el TF (Frecuencia de Término): para los términos gato y perro (0,1 pts)

```
Solución: TF es Número de veces que aparece un Término en un documento,
dividido por el número total de términos en el documento
```

Termino	gato	perro
Documento 1	1/4	0/4
Documento 2	0/4	1/4
Documento 3	1/7	1/7

EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** “El gato come pescado”
- **Documento 2:** “El perro come carne”
- **Documento 3:** “El gato y el perro son amigos”

b) Calcular el IDF (Frecuencia Inversa de Documento): para los términos gato y perro. Nota: $log(3/2) = 0{,}1761; log(3) = 0{,}4771$ (0.1 pts)

EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** "El gato come pescado"
- **Documento 2:** "El perro come carne"
- **Documento 3:** "El gato y el perro son amigos"

b) Calcular el IDF (Frecuencia Inversa de Documento): para los términos gato y perro. Nota: $log(3/2) = 0{,}1761; log(3) = 0{,}4771$ (0.1 pts)

```
 El IDF se calcula como el logaritmo del
número total de documentos dividido por el número de
documentos que contienen el término.
Termino        gato          perro
IDF          log(3/2)      log(3/2)
```

EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** “El gato come pescado”
- **Documento 2:** “El perro come carne”
- **Documento 3:** “El gato y el perro son amigos”

c) Calcular el TF-IDF: para los términos gato y perro. (0.1 pts)

EJERCICIOS

Tienes un conjunto de documentos y deseas calcular el TF (Frecuencia de Término) y el IDF (Frecuencia Inversa de Documento) para cada término en los documentos.

- **Documento 1:** "El gato come pescado"
- **Documento 2:** "El perro come carne"
- **Documento 3:** "El gato y el perro son amigos"

c) Calcular el TF-IDF: para los términos gato y perro. (0.1 pts)

```
El TF-IDF se calcula multiplicando el TF por el IDF.
Termino        gato                    perro
Documento 1    0.25*0.1761=0.0440      0*0.1761=0
Documento 2    0*0.1761=0              0.25*0.1761=0.0440
Documento 3    0.14*0.1761=0.0246      0.14*0.1761=0.0246
```

Clase 14

BIENVENIDOS A

PRIVACIDAD

“El uso de datos personales tiene riesgos”

PRIVACIDAD

Evita que los datos confidenciales de un individuo se vuelvan públicos.

La privacidad de un individuo A es afectada si:

1. El analista aprende B si no cuenta con los datos del individuo A.
2. El analista aprende B + C si agrega los datos de A.

Un analista preserva la privacidad si no importa si sus datos son considerados en el análisis

Que podemos aprender de los datos

SEGURIDAS

Se relaciona con quien puede manipular los datos:

- Confidencialidad: quien puede ver los datos.
- Integridad: quien puede modificar los datos.
- Disponibilidad: qien puedes modificar los datos.
- Resiliencia.

TÉCNICAS DE PRIVACIDAD

Datos sintéticos: son datos generados artificialmente. (lucen como datos originales)

Ejemplo: Anonimización, k-Anonimato, l-diversidad

Respuestas a consultas: datos agregados de alguna manera. (requieren de una consulta en particular)

Ejemplo: Privacidad diferencial.

Nombre	Fecha Nac.	Sexo	Coding Postal
Rashad Arnold	26/02/2018	M	73909
Alyssa Cherry	08/05/2018	M	14890
Myra Ford	11/05/2018	F	58821
Meredith Perry	31/03/2019	F	465113
Aimee Thornton	26/04/2018	F	90825



Nombre	Fecha Nac.	Sexo	Coding Postal
******	26/02/2018	M	73909
******	08/05/2018	M	14890
******	11/05/2018	F	58821
******	31/03/2019	F	465113
******	26/04/2018	F	90825

Nombre	Fecha Nac.	Sexo	Coding Postal
Rashad Arnold	26/02/2018	M	73909
Alyssa Cherry	08/05/2018	M	14890
Myra Ford	11/05/2018	F	58821
Meredith Perry	31/03/2019	F	465113
Aimee Thornton	26/04/2018	F	90825

¿Cuántas personas nacieron en el 2018?



4

+

¿Cuántas personas hay de cada sexo?



M = 2 y F = 3

TIPOS DE ATRIBUTOS



Nombre	Fecha Nac.	Sexo	Coding	Enfermedad
Rashad	26/02/2018	M	73909	Resfrío
Alyssa	08/05/2018	M	14890	Hepatitis
Myra Ford	11/05/2018	F	58821	Hepatitis
Meredith	31/03/2019	F	465113	VIH
Aimee	26/04/2018	F	90825	Bronquitis

Identificador explicito (IdE)	Conjunto de atributos que individualmente identifican de manera explícita al "dueño" de un registro. Por ejemplo, nombre y RUT
Cuasi identificador (CId)	Conjunto de atributos que colectivamente tienen el potencial de identificar al dueño de un registro. Por ejemplo, día de nacimiento y dirección.
Atributos sensibles (AS)	Conjunto de atributos cuyo valor se desea proteger. Por ejemplo, religión, etnia, enfermedad, etc.
Atributos no-sensibles (AnS)	Conjunto de atributos que no tienen un carácter privado (y por lo tanto no se busca proteger). Por ejemplo, equipo de futbol o película favorita.

DE-IDENTIFICACIÓN

Remueve la asociación entre una persona y un conjunto de datos.

Técnicas:

- Supresión (remover los datos)
- Variación (“revolver” los datos)
- Enmascaramiento

Nombre	Fecha Nac.	Sexo	Coding Postal
Rashad Arnold	26/02/2018	M	73909
Alyssa Cherry	08/05/2018	M	14890
Myra Ford	11/05/2018	F	58821
Meredith Perry	31/03/2019	F	465113
Aimee Thornton	26/04/2018	F	90825

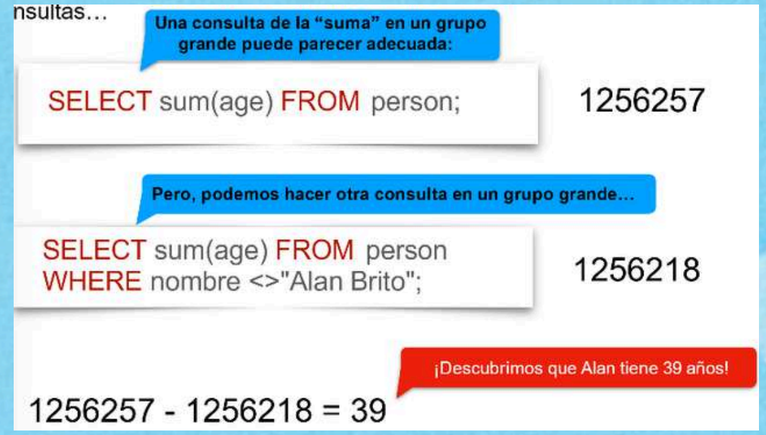
Nombre	Fecha Nac.	Sexo	Coding Postal
******	26/02/2018	M	73909
******	08/05/2018	M	14890
******	11/05/2018	F	58821
******	31/03/2019	F	465113
******	26/04/2018	F	90825

RE-IDENTIFICACIÓN

Ataque de asociación: re-asocia una persona con un conjunto de datos.



Ataque de diferenciación: burla la privacidad haciendo preguntas msivas muy astutas.



K-ANONIMATO

si alguien accede a la base de datos, siempre va a encontrar un "bloque" de al menos k personas que se ven exactamente iguales.

Cada registro es indistinguible de al menos otros k-1 registros respecto de un conjunto de Clds.

Debilidades:

1. Homogeneidad: si conoces los Clds de una persona puedes saber el grupo al que pertenece, y si tiene el mismo valos entonces se puede saber un dato sencible.
2. Data auxiliar: si tienes información adicional externa, y la usas para reducir o eliminar otras opciones.

Zip en el rango de 130**

name	zip	age	nationality
Bob	13053	35	??

+

zip	age	nationality	disease
130**	<30	*	Heart
130**	<30	*	Heart
130**	<30	*	Flu
130**	<30	*	Flu
1485*	>40	*	Cáncer
1485*	>40	*	Heart
1485*	>40	*	Flu
1485*	>40	*	Flu
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer

Los japoneses tiene baja incidencia en enfermedades al corazón

name	zip	age	nationality
Umeko	13068	24	Japan

+

zip	age	nationality	disease
130**	<30	*	Heart
130**	<30	*	Heart
130**	<30	*	Flu
130**	<30	*	Flu
1485*	>40	*	Cáncer
1485*	>40	*	Heart
1485*	>40	*	Flu
1485*	>40	*	Flu
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer
130**	30-40	*	Cáncer

L-DIVERSIDAD

Es una extensión y mejora directa del k-Anonimato, ya que exige que la información sensible dentro de cada bloque sea variada.

1. Homogeneidad: previene ataques.
2. Data auxiliar: incremente la resistencia frente al ataque.

Los japoneses tiene baja incidencia en enfermedades al corazón.

No tiene cancer.

No tiene una enfermedad al corazón.

name	zip	age	nationality
Umeko	13068	24	Japan

+

zip	age	nationality	disease
130**	<30	*	Heart
130**	<30	*	Heart
130**	<30	*	Flu
130**	<30	*	Flu
1485*	>40	*	Cancer
1485*	>40	*	Heart
1485*	>40	*	Flu
1485*	>40	*	Flu
130**	30-40	*	Cancer
130**	30-40	*	Cancer
130**	30-40	*	Cancer
130**	30-40	*	Cancer

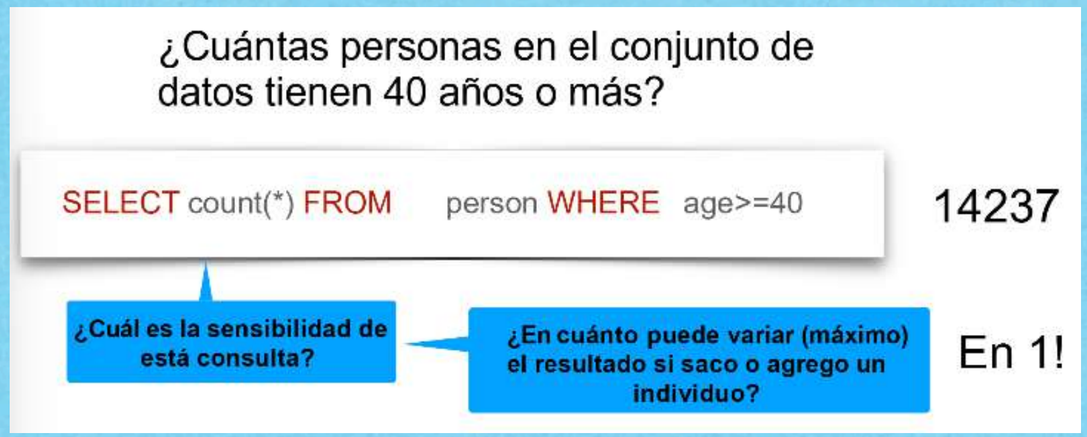
PRIVACIDAD DIFERENCIAL

Garantiza que un algoritmo aleatorio no revele información específica de ningún individuo.

Se logra asegurando que la probabilidad de obtener un resultado particular en casi la misma, independientemente si la persona esta o no en la base de datos.

Solo funciona para respuestas a consultas

Única forma conocida más cercana a una “anonimización verdadera”



COMPOSICIÓN SECUENCIAL

Tenemos n mecanismos:

$F_1(x), \dots, F_n(x)$ → satisfacen $e_1, \dots, e_n$ – privacidad diferencial.

Entonces $G(x) = (F_1(x), \dots, F_n(x))$

→ satisfacen $e_1 + \dots + e_n$ – privacidad diferencial.

COMPOSICIÓN PARALELA

Si $F(x)$ satisfacen e_1 – privacidad diferencial y dividimos el conjunto de datos X en $x_1 \cup \dots x_k = X$.

→ $F(x_1), \dots, F(x_k)$ satisfacen e – privacidad diferencial

EJERCICIOS

Examen 2025 -2

Pregunta 2 – ejercicio d)

Defina cada uno de los siguientes conceptos y cuál es su rol para mantener la privacidad: (5 pts)

a) k-Anonimato: es una técnica para mantener la privacidad de los datos a través de exigir que los cuasi identificadores estén al menos k veces repetidos en un conjunto de datos.

b) l-Diversidad: es una técnic para mantener la privacidad de los datos sensibles en un conjunto de dados anonimizados. Exige que por cada conjunto de cuasi identificadores, existan al menos 1 valores distintos para los datos sensibles.

c) Privacidad diferencial: La privacidad diferencial es un mecanismo que garantiza que un algoritmo aleatorio que opera sobre un conjunto de datos no revele información específica de ningún individuo.

¡GRACIAS!