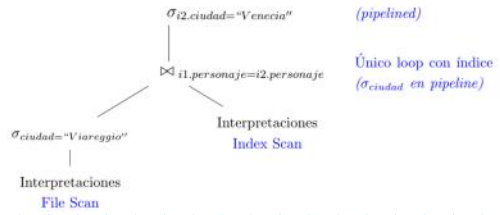
2021 2C Integradora

domingo, 25 de junio de 2023 17:37

(Procesamiento de Consultas) En Italia, el carnaval se celebra en cientas de ciudades con la presencia de personajes típicos o "máscaras". Algunos de estos personajes típicos son Arlequín, Pantaleón o Colombina, por ejemplo. Luego de celebrado el Carnaval 2023, la *Red Italiana de Carnaval* quiere poner en contacto a personas que interpretaron a la misma máscara en distintas ciudades. Para comenzar con este proyecto, se pondrá en contacto a personas que interpretaron al mismo personaje en el carnaval de Venecia y en el carnaval de Viareggio. Para realizar esta tarea, se cuenta con una tabla en una base de datos relacional que registra los datos de las personas que tuvieron algún papel en algún carnaval de 2023:

- `Interpretaciones(apellido_nombre, mail, ciudad, personaje)`
  ('Elsa Rivetti', 'erivetti@gmail.com', 'Venecia', 'Arlequín')

Para obtener el resultado requerido, se utilizará el siguiente plan de ejecución:



La tabla dispone de un índice de clustering de tipo árbol por `personaje`, de altura 3. Además, puede considerar para sus cálculos la siguiente información de catálogo:

INTERPRETACIONES
n(Interpretaciones) = 20.000
B(Interpretaciones) = 2.000
V(ciudad, Interpretaciones) = 20
V(personaje, Interpretaciones) = 100

Se pide:

(a) Estime el costo del plan de ejecución diseñado.

(b) Estime el tamaño del resultado, en términos de cantidad de tuplas.

a) $c(\sigma_1) = B(R)$

$c(\bowtie_1) = n(\sigma_1) \cdot \left(H(I(pj,R)) + \left\lceil \frac{B(R)}{V(pj,R)} \right\rceil \right)$

$n(\sigma_1) = \frac{n(R)}{V(c,R)}$

$c(Query) = c(\sigma_1) + c(\bowtie_1)$

$= B(R) + \frac{n(R)}{V(c,R)} \cdot \left(H(I(pj,R)) + \left\lceil \frac{B(R)}{V(pj,R)} \right\rceil \right)$

b) $n(Query) = \frac{n(\bowtie_1)}{V(c_2, \bowtie_1)} = \frac{n(\bowtie_1)}{V(c,R)} = \frac{n(R)^2}{V(c,R)^2 \cdot V(pj,R)}$

$n(\bowtie_1) = \frac{n(\sigma_1) \cdot n(R)}{\max\left(V(pj,\sigma_1), V(pj,R)\right)}$

$= \frac{n(\sigma_1) \cdot n(\,)}{\max\left(V(pj,R), V(pj,R)\right)}$

$= \frac{\frac{n(R)}{V(c,R)} \cdot n(R)}{V(pj,R)} = \frac{n(R)^2}{V(c,R) \cdot V(pj,R)}$

(NoSQL) Para una colección en *MongoDB* bajo un esquema de *sharding* con replicación, indique si las siguientes afirmaciones son verdaderas o falsas, justificando su respuesta.

(a) Cuando se buscan documentos con un valor específico de un atributo A, se deberá consultar a todos los replica sets, que deberán buscar en su base local. F

(b) Cuando se inserta un nuevo documento en un replica set, se deberá esperar a que el documento se escriba en todos los nodos de ese replica set antes de devolverle el control al usuario. F

(c) Si el sharding se realizó por un atributo A, y luego se hace una consulta por un valor específico de un atributo B, esa consulta será poco eficiente porque cada nodo deberá leer toda su base local en busca de documentos que cumplan con esa condición. V

(d) Cuando se hace una consulta por un valor específico del atributo de sharding, el resultado será un único documento, y el mismo podrá ser hallado en forma muy eficiente por el nodo que lo almacena. F

a) Si la consulta es por el atributo de sharding la consulta ira solo al replica set correspondiente En cambio, si no lo es, si ira a todos los replica sets

b) Es un parametro configurable; puede ser necesario o no que esto ocurra

c) La consulta esta distribuida por todos los nodos Atributo B puede tener indices, pero aun asi se debera buscar en todos los nodos

d) No es necesario que el atributo de sharding sea unico Si sera eficiente pues debera existir un indice.

(NoSQL) Indique si las siguientes afirmaciones sobre los LSM-trees (*log-structured merge trees*), son verdaderas o falsas, justificando brevemente su respuesta.

(a) Cuando se hace un rolling merge entre la estructura C_0 en memoria y la estructura C_1 en disco, los datos de C_0 no se van agregando al C_1 existente, sino que se guardan en una nueva estructura C_1^* en disco. F

(b) Cuando un usuario inserta un par (k, v), el mismo se agrega en C_0 en memoria – asumiendo que haya espacio–, exista o no ya una entrada para k en C_1 en disco. V

(c) La estructura C_0 del LSM-tree es mantenida en memoria volátil. Por lo tanto, el gestor no le garantiza al usuario la durabilidad de los datos que se encuentran escritos en C_0. F

(d) Si se buscan los datos asociados a una clave k en C_0 y no se encuentra ninguna entrada para k, entonces se deberá buscar en el nivel C_1. F

a) C^0 sera agregado a C^1. C^1 sera compactado y unido a C^2 una vez llegue a cierto tamaño

b) LSM tree es append-only y se agregan las diferencias como nuevas entradas

c) C_0 si bien esta en memoria, se garantiza la durabilidad con las reglas WAL y FSL

(c) La estructura C_0 del LSM-tree es mantenida en memoria volátil. Por lo tanto, el gestor no le garantiza al usuario la durabilidad de los datos que se encuentran escritos en C_0. F

(d) Si se buscan los datos asociados a una clave k en C_0 y no se encuentra ninguna entrada para k, entonces se deberá buscar en el nivel C_1. F

(e) La estructura C_0 en memoria conviene implementarla con un B-tree. F

c) C_0 si bien esta en memoria, se garantiza la durabilidad con las reglas WAL y FSL

d) Se utiliza un bloom filter primero, para saber si vale la pena buscar

e) Al estar en memoria, un arbol binario balanceado es mas simple

(Concurrencia y Transacciones) Complete el siguiente cuadro referente un solapamiento *tentativo* de tres transacciones bajo el esquema de control de concurrencia basado en timestamps, indicando en cada fila qué cambios se producen en los `read_TS` y `write_TS` a partir de la operación respectiva. Detenga el análisis cuando encuentre que una transacción deberá ser abortada con el fin de garantizar la serializabilidad, indicando de qué transacción y operación se trata, y cuál es la regla que viola. Considere que los timestamps de las transacciones coinciden con sus subíndices.

A modo de ejemplo se ha completado la primera fila, $R_{T_6}(Y)$.

	T_6	T_8	T_9	X	Y	Z
Valores inic.	TS=6	TS=8	TS=9	read_TS=0 write_TS=0	read_TS=0 write_TS=0	read_TS=0 write_TS=0
	$R(Y)$			–	read_TS(Y)=6	–
		$W(X)$		write_TS= 8		
			$W(Z)$			WRITE_TS = 9
			$R(X)$	WRITE_TS = 9		
		$R(Z)$				READ-TS = 8

READ TOO LATE → Lee item modificado por trans posterior

se aborta T8 para rehacerla

T8 ⇄ T9

(Recuperación) Un SGBD implementa el algoritmo de recuperación UNDO con checkpoint activo. Luego de una falla, el sistema encuentra el siguiente archivo de log:

```
01 (BEGIN, T1);                 08 (WRITE T3, A, 8);
02 (BEGIN, T2);                 09 (WRITE T2, B, 15);
03 (WRITE T1, A, 13);           10 (COMMIT, T2);
04 (WRITE T2, C, 7);            11 (END CKPT);
05 (COMMIT, T1);                12 (BEGIN, T4);
06 (BEGIN CKPT, T2);            13 (WRITE T3, B, 8);
07 (BEGIN, T3);                 14 (WRITE T4, C, 9);
```

Explique cómo se llevará a cabo el procedimiento de recuperación, indicando hasta qué punto del archivo de log se deberá retroceder, y qué cambios deberán ser realizados en disco y en el archivo de log.

Trans con commit: T_1, T_2
Trans sin commit: T_3, T_4

Se retrocede hasta BEGIN CKPT
T_2 se ignora, esta en disco y con commit antes de BEGIN CKPT

14 - C=9
13 - B=9
08 - A=8

Escribir abort T3, abort T4 en el log