**Práctica 3 – Índices**

Juan Manuel Cardeñosa Borrego

**Ejercicio 2**

-- Creamos la tabla con los atributos

CREATE TABLE PRUEBA(

CLAVE number(16,0) primary key, --Tiene numeros de hasta 16 cifras y 0 decimales

DISPERSO number(16,0),

CONCENTRADO number(16,0),

IDISPERSO number(16,0),

ICONCENTRADO number(16,0),

BCONCENTRADO number(16,0)

);

-- Rellenar con datos de la tabla

DECLARE

I NUMBER(16,0);

R NUMBER(16,0);

BEGIN

FOR I IN 1..500000 LOOP

R := DBMS\_RANDOM.VALUE(1,1000000000);

INSERT INTO PRUEBA VALUES(I, R, MOD(R,11), 1000000000-R, MOD(1000000000-R, 11),

MOD(2000000000-R, 11));

END LOOP;

END;

/

-- No olvidar de hacer el commit para confirmar los cambios

COMMIT;

**Ejercicio 3**

-- Crear tres índices sobre la tabla

CREATE INDEX PID ON PRUEBA (IDISPERSO);

CREATE INDEX PIC ON PRUEBA (ICONCENTRADO);

CREATE BITMAP INDEX PBC ON PRUEBA (BCONCENTRADO);

**Ejercicio 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SELECT COUNT(\*) FROM PRUEBA WHERE | CLAVE = 50000; | DISPERSO = 50000; | CONCENTRADO = 5; | IDISPERSO = 50000; | ICONCENTRADO = 5; | BCONCENTRADO = 5; |
| PLAN | INDEX UNIQUE SCAN| PRUEBA\_PK | TABLE ACCESS FULL| PRUEBA | TABLE ACCESS FULL| PRUEBA | INDEX RANGE SCAN| PID | INDEX RANGE SCAN| PIC | BITMAP INDEX SINGLE VALUE| PBC |
| A-Time | 00:00:00.01 | 00:00:00.06 | 00:00:00.05 | 00:00:00.01 | 00:00:00.01 | 00:00:00.01 |
| Reads | 24 | 2260 | 1949 | 3 | 87 | 12 |
| “consistent gets | 109 | 2452 | 2506 | 173 | 265 | 179 |
| “physical read total bytes | 532480 | 19013632 | 19021824 | 2973696 | 3768320 | 3072000 |

**Ejercicio 5**

Cuando se usan los índices, van mucho más rápido, siendo el número de “consistent gets” es menor, accediendo a menos número de bloques y, por consiguiente, se hace un uso mayor de memoria.

Una búsqueda por índices va a ser siempre mucho más eficiente en tiempo y recursos que las que acceden dato por dato hasta encontrar el dato a consultar.

**Ejercicio 6**

UPDATE PRUEBA SET DISPERSO = DISPERSO + 7; // 12 segundos

UPDATE PRUEBA SET IDISPERSO = IDISPERSO + 7; // 17 segundos

Tarda más el de “IDISPERSO” ya que tiene que modificar el índice entero, mientras que en el de “DISPEROSO” solo hace la suma.

**Ejercicio 7**

Vaciamos la caché previamente.

SELECT COUNT(\*) FROM PRUEBA WHERE IDISPERSO BETWEEN 10000 AND 20000;

Hay que observar que se ha hecho una operación denominada INDEX RANGE SCAN, ya que estamos realizando una búsqueda por un rango y el índice está ordenado se puede realizar la búsqueda del rango usando el propio índice.

**Ejercicio 8**

Vaciamos la caché previamente.

SELECT COUNT(\*) FROM PRUEBA WHERE IDISPERSO+ICONCENTRADO BETWEEN 10000 AND 20000;

Ahora se hace un TABLE ACCESS FULL, y se ha tardado 9 veces más en realizar dicha instrucción. Además de muchísimas más lecturas de bloque en RAM (consistent gets).

**Ejercicio 9**

Creamos el índice que nos indican.

CREATE INDEX FIX ON PRUEBA(IDISPERSO+ICONCENTRADO);

**Ejercicio 10**

Vaciamos la caché previamente.

SELECT COUNT(\*) FROM PRUEBA WHERE IDISPERSO+ICONCENTRADO BETWEEN 10000 AND 20000;

**Ejercicio 11**

Ahora el coste se reduce de forma considerable ya que se realiza la operación INDEX RANGE SCAN y no un TABLE ACCESS FULL, haciendo que el número de lecturas de bloque se reduzca.

**Ejercicio 12**

ALTER SYSTEM FLUSH SHARED\_POOL;

ALTER SYSTEM FLUSH BUFFER\_CACHE;

UPDATE PRUEBA SET IDISPERSO = IDISPERSO + 7;

**Ejercicio 13**

Ha tardado el doble de tiempo y se han realizado 4 veces más lecturas a bloques que antes.