데이터베이스 개론 (SWE 3003-41, 2021 가을학기)

담당강사: 이상원

학번: _____2017313107_____ 이름:____이승태____ 점수:_____/100

1. 데이터 페이지 접근의 시간적 지역성(temporal locality)를 설명하고, LRU 버퍼 교체정책과 그 관련성을 논하시오. (각 3 점, 총 5 점)

시간적지역성:
어떤 페이지를 접근했을 때, 다시 그 페이지를 접근할 확률이 높다.

LRU 정책과 관련성:
LRU 는 가장 오랫동안 사용되지 않은 페이지를 victim page 로 선정하는 알고리즘으로, 최근 접근한 페이지를 다시 접근할 가능성이 많기 때문에 temporal locality 를 잘 생각한 알고리즘이라고 할 수 있다.

2. 5 분에 한번씩 접근되는 8KB 크기의 페이지 P 가 있다. 그런데, DRAM 가격은 10$/GB 이고, 100$짜리 하드디스크는 8KB 페이지 단위로 초당 200 IOPS 를 제공하고, 1000$ 플래시메모리 SSD 는 초당 50,000 IOPS 를 제공한다. 페이지 P 를 DRAM + HDD 상에서는 어디에 위치시키는 것이 경제적인지와, DRAM + SSD 에서는 어디에 위치 시키는 것이 경제적인지 간단한 수식으로 설명하시요.? HDD 와 SSD 의 용량은 $/GB 는 고려에서 제외함. (각 5 점, 총 10 점)

DRAM + HDD: (DRAM) 유리
이유: HDD 는 IOPS 가 느리므로 5 분마다 한 번씩 접근되는 page 는 hot 한 page 라고 할 수 있다. 따라서 DRAM 에 올려야 한다.

DRAM + SSD: (SSD) 유리
이유: SSD 는 IOPS 가 매우 빠르므로 5 분마다 한 번씩 접근되어도 비교적 cold 한 page 라 판단할 수 있다. 따라서 SSD 에 올리는 것이 더 바람직하다.

3. 버퍼교체정책의 목적은 hit ratio 증가 (다시 말해 miss ratio 감소)이다. hit ratio 가 90%인 특정 OLTP DB 시스템에서는 초당 200 IOPS 를 제공하는 개당 10 만원짜리 하드디스크 10,000 개를 이용해서 시스템을 balanced 상태 (즉, CPU 와 IO 시스템이 모두 100% 가동)로 유지하고 있다 (A 옵션). 한편, 버퍼교체정책을 수정해서 hit ratio 을 95%로 증가시켰고, 스토리지를 개당 천만원, 500,000 IOPS 를 제공하는 SSD 로 교체해서 마찬가지로 balanced 상태로 유지하고자 한다. (B 옵션, 스토리지 용량은 상관없다고 가정하자.)
이때, A, B 옵션에서 스토리지 비용은 각 얼마인가? (각 3 점, 총 5 점)

A 옵션: 10 억 원
(이유: 10 만원짜리 하드디스크를 10000 개 가지고 있으므로 100000 * 10000 = 10 억이 된다.)

B 옵션: 2000 만 원
(이유: 95%로 hit ratio 를 증가시키면, 사실상 성능은 2 배정도 증가한다고 봐도 무방하다. 왜냐하면 10%에서 5%로 miss rate 가 줄었으므로, 사실상 IO 는 5%만큼만해줘도 된다는 이야기이다. A 에서 초당 200IOPS 를 제공하고 10000 개의 하드디스크가 존재하므로 2000000IOPS 가 일어난다. 하지만 hit ratio 가 늘어났으므로 1000000IOPS 만 필요하게 된다. 이를 맞추려면, 스토리지는 2 개 필요하게 되고, 2 천만원이 들게 된다.)

4. 다음 table skewed 에 대해 Q1 질의가 주어졌을 때, 오라클 DBMS 는 결과 튜플을 몇 개로 예측

하는지 대략 설명하시오. 그리고, 실제 조건을 만족하는 튜플은 몇 개인가? 예측치와 실제치가 차이가 나는 이유는 무엇인가? 이를 보정해서 Oracle 이 더 정확한 예측을 할 수 있는 방법에 대해 논하시오. (각 3 점, 모두 맞으면 10 점)

```
create table skewed (a int, b int, c varchar2(650));

BEGIN
    FOR i IN 1..100 LOOP
        FOR j IN 1..i LOOP
        INSERT INTO skewed (a, b, c) values (i, i, rpad('X', 650, 'X'));
        END LOOP;
    END LOOP;
END;

analyze table skewed compute statistics;

Q1: select * from skewed where a between 96 and 100 and b between 96 and 100;
```

예측치: (252.5) 실제 튜플 개수: (15)

차이나는 이유:
Oracle 은 data 가 uniform 하게 분포되어 있다고 가정하고 결과값을 계산하기 때문이다.

보정방안:
Histogram 을 그려주어 데이터에 대한 분포를 알려준다.

5. (13 장 External merge sort) "blocked IO" 최적화는 merge 단계에서 각 런에서 한꺼번에 b 페이지를 읽어서 read IO 시간을 줄이고자 한다. 하드디스크의 경우, 페이지 개수에 무관하게 한번의 연속된 b 페이지 읽기 또는 쓰기에 걸리는 시간을 10ms 라고 가정하자. 이때, b 의 값에 따른 merge 과정의 IO 시간에 있어 trade-off 대해 간단히 논하시오. 반면 flash memory SSD 의 경우 IO 시간이 b 개수 비례한다고 가정하면 위 trade-off 에 어떤 영향을 미치나? (각 5 점, 총 10 점)

정답:
Sort merge 의 step 수는 $\log_{B/b} N/B$ 로 b 값이 증가하면 따라서 증가하고, b 만큼 한 번에 읽어들이므로 N/b 만큼으로 IO 가 감소하게 된다. 따라서 step 수는 b 가 커짐에 따라 증가하고, N/b 는 b 가 증가할수록 작아지므로 이 두 값을 곱했을 때 가장 작은 b 값을 사용하는 것이 좋다.

IO 시간이 b 개수에 비례하면, N/b 의 factor 가 N 으로 바뀌어 결과적으로 그냥 b=1 인 상태가 가장 best 인 상태가 된다.

6. 실습교안에서 설명한 방식대로 오라클 DBMS 에 저장된 두 테이블 TEST, SMALL 에 대해 (두 테이블 공히 한 페이지 (8KB)에 10 개의 Record 를 저장하고, IO 단위는 8KB 페이지로 가정), 질의 Q1 주어졌을 때, 만일 옵티마이저가 실행계획을 tuple-oriented and INDEX-based NESTED LOOP 조인 방법을 (SMALL 이 outer table 로 사용됨) 선택할 경우 동작 방식을 설명하시오. 그리고, 테이블 (heap file)의 페이지 IO 는 몇 번 발생하는지와 그 이유를 간단히 설명하시오. (단, 인덱스 페이지 접근은 제외하시오. 또한, 버퍼 크기가 작아 TEST 테이블 전체를 캐싱할 수 없을 때와 모두 캐싱될 때를 구분해서 답을 제시하시오.) (각 3 점, 총 10 점)

```
create table TEST (a int, b int, c varchar2(650));
BEGIN
    For I in 1..100 LOOP
    FOR j IN 1..10000 LOOP
        INSERT INTO TEST (a, b, c) values (j, j, rpad('X', 650, 'X'));
    END LOOP;
END;
create index test_idx_a on test(a);

create table SMALL (a int, b int, c varchar2(650));
BEGIN
    FOR i IN 1..10000 LOOP
            INSERT INTO SMALL (a, b, c) values (i, i, rpad('X', 650, 'X'));
    END LOOP;
END;
create index small_idx_a on test(a);

Q1: select R.b+S.a
    from test R, small S
    where R.a = S.a;
```

1. 동작방식

Tuple oriented 는 어떤 table 의 한 tuple 에 대해 다른 table 의 값들을 모두 비교하여 같은 값이 존재하면 join 시키게 된다. Index-based nested loop 는 한 tuple 에 대해 다른 tuple 의 값들이 같은지 비교할 때, index 를 타서 값이 같은 것들을 찾는다.

2. 버퍼가 충분히 클 때: (101000) IOs

   설명: 버퍼가 충분히 클 경우에는 TEST table, small table 을 모두 버퍼에 올려놓고 join 을 시키면 되므로 buffer 에 table 을 올리기만 하면 된다. TEST table 의 page 개수는 1 페이지에 10 개의 tuple 이 올라온다고 가정하면 100000page 가 된다. Small table 도 마찬가지로 계산하면 1000page 가 되어서 총 101000 의 IO 가 들게 될 것이다.

3. 버퍼가 작을 때: (10 억) IOs

   설명: 버퍼가 작을 때에는 버퍼에 TEST table 일부를 올리고 필요 없는 부분은 내리는 방법을 택해야한다. Small table 의 tuple 하나당 test table 을 다 훑어봐야 하므로, 대략적으로 (small table 의 tuple 의 수) * (test table 의 page 수) 만큼의 IO 가 일어날 것이다. 10000 * 100000 = 1000000000(십억)IO 가 들게 된다.

7. 두 트랜잭션 T1, T2 가 아래 순으로 SQL 수행시, T2 가 아래 4 가지 다른 Isolation level 로 동작할 때, 각 문장의 수행순서 (예: 1->2->3->4->5), T2 4 번 결과값, 그리고 어떤 lock 을 언제 획득/release 하는지 설명하시오. (각 5 점, 총 20 점)

create table account (id number, balance number);

insert into account values (1, 100);

commit;

T1	T2
2 update account set balance = balance + 10 where id = 1; 3 Commit;	1 Select * from account where id = 1; 4 Select * from account where id = 1; 5 Commit

a) 교안 16.4, 16.6.3 절에 배운 Single-version 지원 DBMS "READ COMMITTED" 모드일 때

순서: (1->2->3->4->5) T2 4 결과값: (1, 110)

Lock 획득/release 설명:

READ COMMITED mode 에서는 read lock 을 짧게 잡게 된다. 1, 4 에서 값을 읽을 때 잠깐 잡고 풀어준다. T1,2 에서는 write lock 을 짧게 잡고 commit 될 때, 풀어주게 된다.

b) Single-version 지원 DBMS "REPEATABLE READ" 모드:

순서: (1->4->5->2->3) T2 4 결과값: (1, 100)

Lock 획득/release 설명:

REPEATABLE READ 모드는 read lock 을 길게 잡게 된다. 그러므로 1 에서 id=1 값을 읽었으므로 commit 할 때까지 잡게 되고, T1 은 commit 이 될 때까지 기다려야 한다. T1, 2 에서도 write lock 을 잠깐 잡고 commit 하는 순간 풀어주게 된다.

오라클 "READ COMITTED" 모드: 순서: (1->2->3->4->5) T2 4 결과값: (1, 110)

Lock 획득/release 설명:

오라클은 read 할 때 lock 을 잡지 않는다. 하지만 T1, 2 에서 write lock 을 잠깐 잡아주었다가 풀어준다.

오라클 "SERIALIZABLE" 모드: 순서: (1->2->3->4->5) T2 4 결과값: (1, 100)

Lock 획득/release 설명:

오라클은 read 할 때 lock 을 잡지 않는다. 하지만 T1, 2 에서 write lock 을 잠깐 잡아주었다가 풀어준다.

8. DBMS 회복관리자(recover manager)의 주요 역할은 트랜잭션 원자성(atomicity)과 지속성(durability)를 보장하는 것이다. 아래 질문에 답하시오. (총 15 점)

1. 원자성과 지속성에 대해 설명 하시오. (각 3 점, 총 5 점)

원자성:
Transaction 을 수행할 때, all or nothing 을 보장하는 property 로 transaction 중간에 종료되면, 이전상태로 되돌리는 nothing 을, commit 되면 그 값을 보장하는 all 을 보장해야 한다.
지속성: 한번 commit 되면 database 상에 영구적으로 적용되어야하는 property 이다.

2. 예를 들어, 트랜잭션 T 가 2 페이지 P1, P2 를 갱신한 후 commit 시 force 정책을 사용하는 경우, 스토리지에 있는 P1, P2 의 이전 버전을 새 버전으로 덮어쓰는 과정에서 failure 가 날 수 있고, 이때 시스템 재 시작 후 스토리지 DB 는, P1 에 대한 atomic write 가 보장되지 않을 수도 있고, 또는 P1 만 성공적으로 덮여쓰였고, P2 는 이전 버전 상태로 있을 수도 있다. 이러한 failure 에 대비해서, 원자성과 지속성 보장을 구현하는 방안을 한 가지 제시하시오. (10 점)

No steal 정책이 있다. No steal 은 commit 되기 전까지 buffer 에 계속 남겨두는 정책이다. 이 정책을 사용할 경우, 만약 transaction 중간에 system 이 crash 날 경우, 어차피 disk 에 덮어써지면 안되므로 buffer 의 data 가 지워져도 상관이 없다. 그러므로 atomicity 의 all or nothing 에서 nothing 을 보장해주게 된다.

9. Write-Ahead Log (WAL)이란? 왜 필요한가? 그리고, 로그버퍼를 비휘발성메모리 (전원이 나가도 데이터가 그대로 남아있는 메모리) 에 관리하는 경우에도 WAL 프로토콜을 여전히 따라야 하는지 논하시오. (각 3 점, 총 10 점)

- WAL 이란: 어떤 page 를 disk 에 써주기 이전에 undo log 를 log device 에 써주는 protocol 을 의미한다.
- 필요성: 만약 이 프로토콜을 사용하지 않는다면 undo log 가 log buffer 에 있는 상태에서 crash 났을 경우, 이 정보들이 날라가므로, disk 에 있는 page 가 덮어씌워져서 원래대로 되돌려놔야 하는 page 알 방법이 없게 된다.
-
- 필요여부 및 원인: 필요 없다. Log device 에 써주는 것은 log buffer 에 있는 내용들이 시스템이 crash 가 날 경우 다 없어지게 되므로, 다시 undo 할 방법이 없어 나온 protocol 이기 때문이다. 비휘발성메모리를 사용하면 log 가 남아있으므로, wal protocol 이 필요 없게 된다.

10. 체크포인트란 무엇이고, 왜 필요한지 논하시오. (각 3 점, 총 5 점)

WHAT?
체크포인트는 주기적으로(몇 분마다) buffer 에서 disk 로 flush 해주는 기능이다.

WHY?
만약 시스템이 crush 가 났을 때, 복구 속도를 높여줄 수 있다.

---------- 끝! 시험 준비하느라 수고 많았습니다. ----------