3. 그림으로 공부하는 오라클 구조 - Ch3. 캐시와 공유 메모리

캐시(Cache)

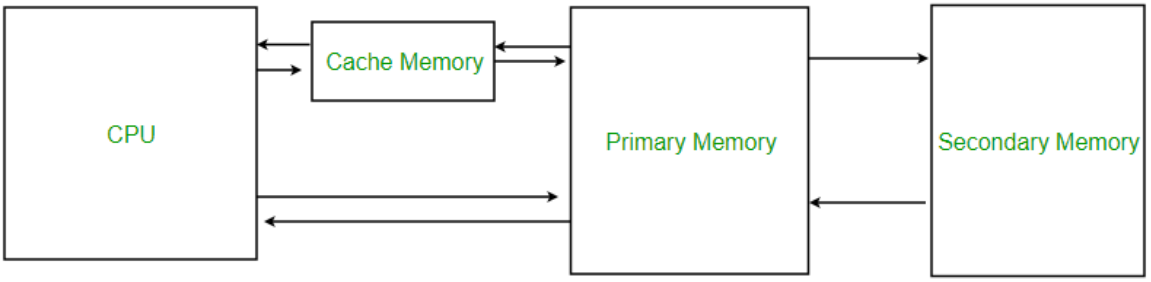
- 오라클 시스템 구조에서의 캐시를 살펴보기 이전에 컴퓨터 공학에서 사용되는 캐시라는 용어에 대한 일반적인 개념부터 살펴보는 게 좋다

컴퓨터 구조에서의 캐시 개념

https://www.geeksforgeeks.org/cache-memory-in-computer-organization/

*Cache Memory is a special very **high-speed memory**. The cache is a smaller and faster memory that **stores copies of the data from frequently used main memory locations**. There are various different independent caches in a CPU, which store instructions and data. The most important use of cache memory is that it is used to reduce the average time to access data from the main memory.*

- 캐시는 캐시 메모리라고 불리며 빠른 속도의 메모리이다. 캐시 메모리는 메인 메모리에서 자주 사용되는 데이터를 더 빠르게 접근하기 위해 따로 저장해두는 메모리를 뜻한다.



CPU와 메모리 사이에 위치한 캐시 메모리

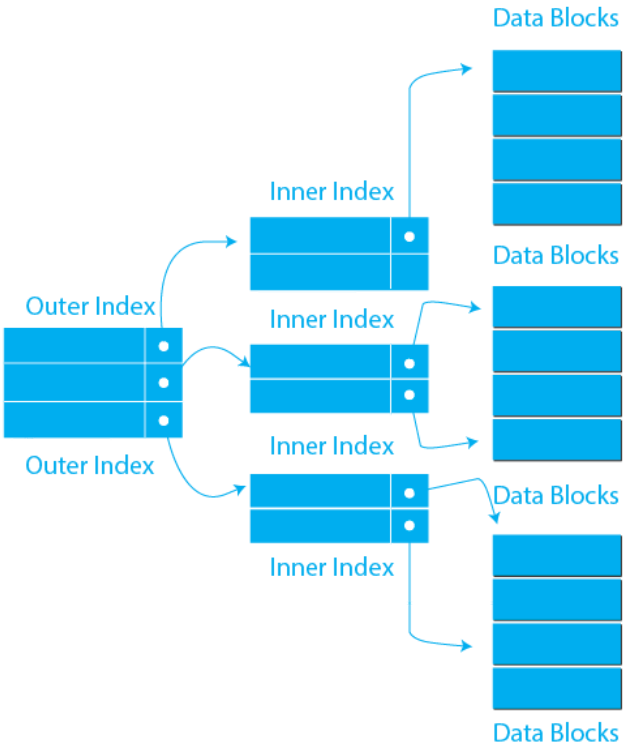
- 위는 컴퓨터 구조에서의 캐시 개념을 설명한 것이다. 일반적으로 캐시는 '자주 쓰이는 데이터를 더 빠르게 접근하도록 임시보관하는 메모리 공간'이라고 생각하면 된다.
- 오라클 시스템에서의 캐시를 살펴보기 이전에 일반적인 캐시 개념을 이용해 유추해보자면 **시간이 상당히 걸리는 디스크 접근 횟수를 줄이기 위해 자주 쓰이는 데이터를 캐시라는 메모리 공간에 두고 사용하지 않을까**라는 추측을 할 수 있다.

오라클의 캐시(데이터 캐시 혹은 버퍼 캐시)

- 오라클의 캐시는 data cache 혹은 buffer cache라고 불린다. 서버 프로세스(클라이언트의 요청을 처리하는 오라클 프로세스)가 원하는 데이터가 캐시에 존재하면 그 데이터에 접근하여 DISK I/O를 줄인다. 즉, SQL의 처리 속도를 높인다.

오라클 캐시의 데이터 관리 단위 : 블록

- 오라클은 'block'이라는 단위로 데이터를 관리한다. 디스크 I/O 및 버퍼 캐시 모두 블록 단위로 관리한다.(OS에도 블록이라는 개념이 있지만 이것과는 다르다)
- 데이터를 한 개만 꺼내오려고 해도 필요한 데이터를 포함한 블록 자체가 캐시에 보관된다. 2KB, 4KB, 8KB, 16KB, 32KB 등 블록 사이즈를 고를 수 있다.

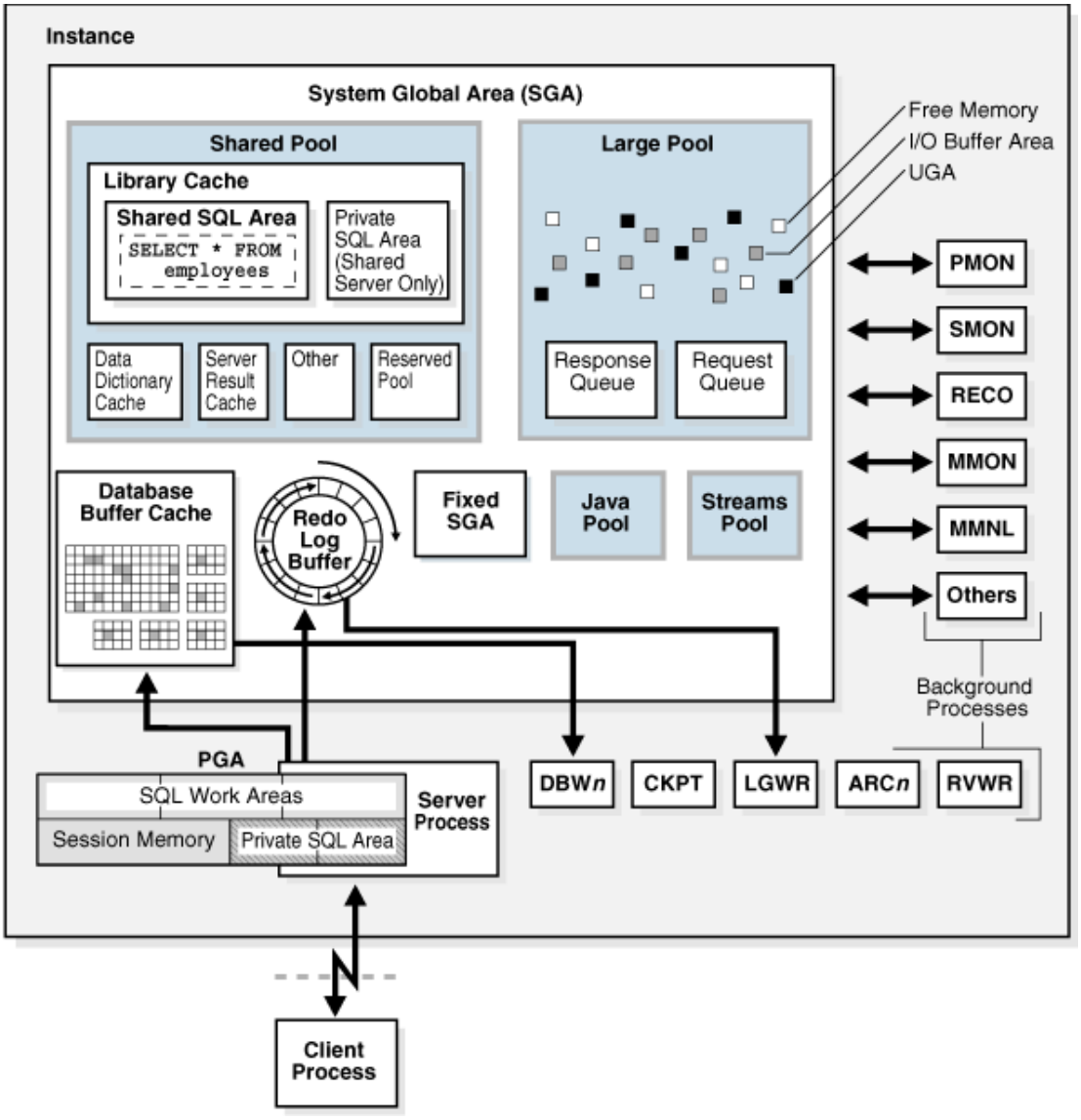


Multi Level Index 구조

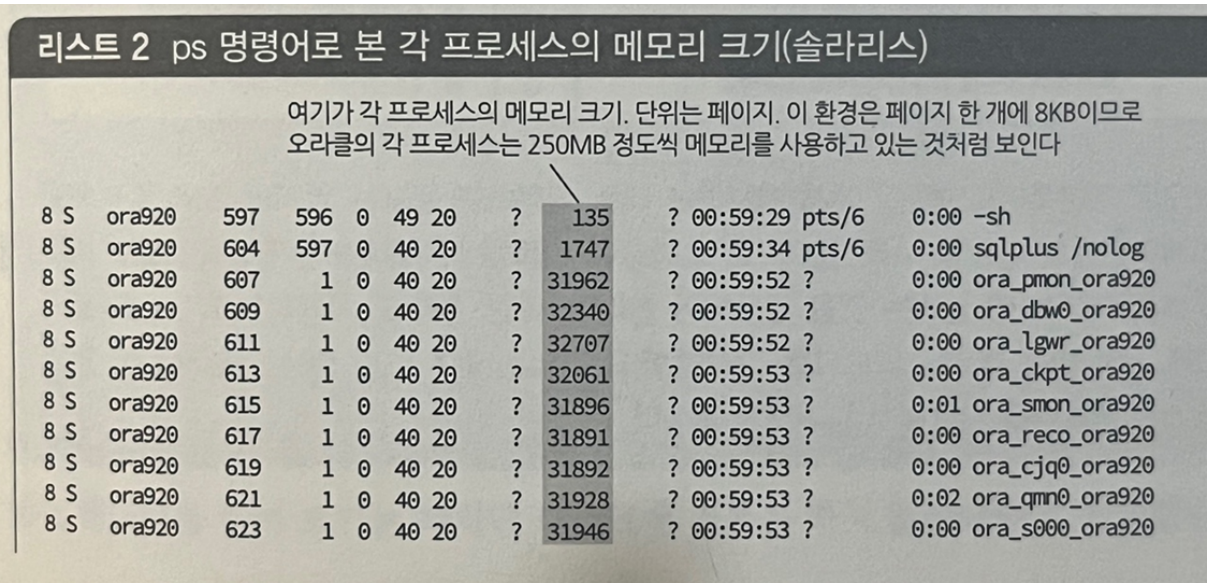
- 그림과 같이 인덱스가 단계적으로 되어 있을 때 인덱스를 이용해 데이터를 구하기 위해서는 Outer인덱스의 블록에서 데이터를 가져와(DISK IO 1회) Inner 인덱스 블록을 찾으러 다시 디스크에 접근한다(DISK IO 2회). 그리고 나서 데이터의 주소를 가져와 데이터 블록에 다녀온다.(DISK IO 3회). 위와 같은 구조에서는 디스크에 총 3번을 접근해야 원하는 데이터에 접근할 수 있다.
- 이 때, 원하는 데이터가 데이터 캐시(버퍼 캐시)에 존재한다면, DISK에 접근을 한 번도 하지 않고 디스크 접근이라는 물리적인 움직임 없이 전기적인 신호만으로 클라이언트의 요청에 대한 응답이 가능하다. 이것이 캐시를 이용하는 이유이다.

공유 메모리(SGA;System Global Area)

- 프로세스는 캐시를 공유한다. 기본적으로 다른 프로세스의 메모리를 보는 것을 불가능하다. 데이터에 손상을 입히지 않도록 OS가 제한을 두기 때문이다. DBMS는 이런 불편한 점을 해결하고 다른 프로세스와 데이터를 공유하기 위해 OS의 기능 중 특수한 메모리 기능인 '공유 메모리'를 사용한다. 즉 자신의 캐시를 다른 프로세스도 볼 수 있게 되고 자신 또한 다른 프로세스의 캐시를 볼 수 있게 된다. 이런 공유 메모리 영역을 오라클에서 SGA(System Global Area)라고 불린다. 공유되지 않는 메모리는 PGA(Program Global Area, 이 명칭은 Process가 Program이 실행된 형태를 의미하는 것을 안다면 쉽게 와닿는다)
- 메모리의 데이터를 공유하기 떄문에 데이터에 손상을 가하지 않도록 락을 걸어 배타 제어(exclusive control)을 DBMS가 내부적으로 수행한다. 예를 들어, 동시에 같은 데이터를 변경하려고 하는 등의 시도를 발생하지 않도록 방지하는 것이다.



오라클 시스템 구조



리스트 2 ps 명령어로 본 각 프로세스의 메모리 크기(솔라리스)

여기가 각 프로세스의 메모리 크기. 단위는 페이지. 이 환경은 페이지 한 개에 8KB이므로 오라클의 각 프로세스는 250MB 정도씩 메모리를 사용하고 있는 것처럼 보인다

```
8 S   ora920   597   596  0  49 20   ?    135   ? 00:59:29 pts/6    0:00 -sh
8 S   ora920   604   597  0  40 20   ?   1747   ? 00:59:34 pts/6    0:00 sqlplus /nolog
8 S   ora920   607     1  0  40 20   ?  31962   ? 00:59:52 ?        0:00 ora_pmon_ora920
8 S   ora920   609     1  0  40 20   ?  32340   ? 00:59:52 ?        0:00 ora_dbw0_ora920
8 S   ora920   611     1  0  40 20   ?  32707   ? 00:59:52 ?        0:00 ora_lgwr_ora920
8 S   ora920   613     1  0  40 20   ?  32061   ? 00:59:53 ?        0:00 ora_ckpt_ora920
8 S   ora920   615     1  0  40 20   ?  31896   ? 00:59:53 ?        0:01 ora_smon_ora920
8 S   ora920   617     1  0  40 20   ?  31891   ? 00:59:53 ?        0:00 ora_reco_ora920
8 S   ora920   619     1  0  40 20   ?  31892   ? 00:59:53 ?        0:00 ora_cjq0_ora920
8 S   ora920   621     1  0  40 20   ?  31928   ? 00:59:53 ?        0:02 ora_qmn0_ora920
8 S   ora920   623     1  0  40 20   ?  31946   ? 00:59:53 ?        0:00 ora_s000_ora920
```

책 참고 이미지

버퍼 캐시를 정리하는 LRU 알고리즘

- 정보처리기사에서 페이징 처리 알고리즘에서 배웠던 LRU 알고리즘이 버퍼 캐시를 정리하기 위해 쓰인다. 간단하게 최근에 사용하지 않은 데이터부터 캐시 아웃(버리기)하는 것이다.

데이터 변경도 캐시에서 이루어진다

- 데이터를 변경(update)하는 것과 같은 요청 또한 캐시에서 선제적으로 이루어진다. 설명하자면 클라이언트의 데이터 변경(UPDATE)요청 시 서버 프로세스는 디스크에 직접 변경된 데이터를 전달하여 기록하지 않고 버퍼 캐시에 읽어온 데이터를 변경하여 놔둔다. 이러면 디스크 IO없이 UPDATE SQL요청을 빠르게 처리하고 클라이언트에게 작업 완료 응답을 보낼 수 있다.
- 서버 프로세스가 캐시에 변경된 데이터를 두면 백그라운드 프로세스인 DBWR가 캐시에서 변경된 데이터가 버려지기 전에 디스크에 기록한다. 디스크에 부하를 주지 않기위해 정기적으로 수행한다.

모든 것을 버퍼 캐시에 두지는 않는다

- 오라클은 큰 테이블이라고 판단하면 풀 스캔한 것을 버퍼 캐시에 오랜 시간 보관하지 않도록 하고 있다. 일반적으로 풀 스캔 시의 데이터는 버퍼 캐시에 적재되지 않는다고 생각하면 된다.

- 큰 테이블, 작은 테이블 등의 기준은 '_small_table_threshold' 과 같은 설정값을 바꾸어 조정할 수 있다.

스토리지 캐시와 OS의 가상메모리

스토리지라는 저장장치의 캐시를 이용하면 IO 시간을 줄일 수 있다

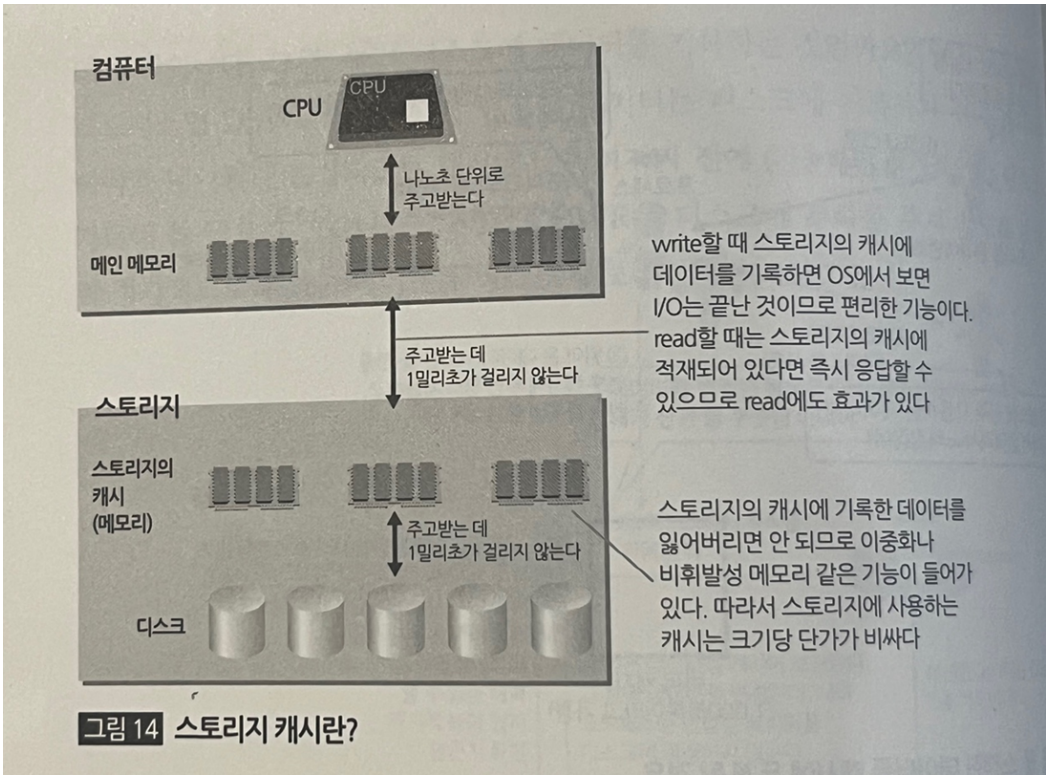


그림 14 스토리지 캐시란?

스토리지 캐시 사용 시 구조

- 스토리지의 캐시에만 데이터 CRUD를 하면 OS입장에서는 스토리지가 캐시 데이터를 알아서 디스크에 기록할 테니 작업이 끝난 것으로 간주할 수 있다. 즉, DBMS입장에서도 디스크 IO가 실제로 이루어지지 않고 스토리지의 캐시에만 기록을 적재한 것이지만 디스크 IO가 끝난 것으로 간주하여 응답을 빠르게 할 수 있다.
- 이렇듯 서버 프로세스가 버퍼 캐시를 사용하는 것만이 데이터 처리의 효율성을 증가시키는 것이 아니라 스토리지 레벨에서도 효율성을 증가시키는 방법이 있다는 것을 알 수 있다.

가상 메모리 때문에 버퍼 캐시가 항상 속도를 증가시키는 것이라고 단언할 수 없다.

- 가상 메모리란 메모리에서 사용 빈도가 낮은 데이터를 물리적인 디스크에 보관하고 메모리에 있는 것처럼 간주하는 기술이다. 이는 OS 레벨에서의 기능이기 때문에 DBMS는 이를 고려하지 않을 것이다.

- 그렇다면 버퍼 캐시가 사용 가능한 물리 메모리의 용량보다 크게 설정되어 있다면 어떻게 될까?
- 그렇게 되면 OS는 가상메모리 기술을 사용해 버퍼 캐시가 사용할 초과분의 용량을 디스크에 마련해 이를 메모리처럼 사용하게 할 것이다.
- 그리고 이 가상 메모리 공간에는 버퍼 캐시에서 잘 사용되지 않는 데이터를 보관하게 하는 것이 가장 효율적일 것이다. 그러다 서버 프로세스가 가상 메모리에 있는 데이터를 요청하면 어떻게 될까?
- 가상 메모리는 디스크에 있기 때문에 디스크 IO가 일어날 것이다. 그러면 서버 프로세스 입장에서는 캐시에만 접근해서 데이터를 가져왔는데도 불구하고 디스크 IO가 일어난 것과 같은 속도로 처리되어 캐시를 사용하는 의미가 없어지게 된다. 즉 아무런 효과도 없는 현상이 발생한다. 가상 메모리 개념과 버퍼 캐시의 개념이 서로 상충하기 때문이다.
- 최근에는 메모리 값이 저렴해지고 용량이 커졌기 때문에 가상 메모리를 사용하지 않는 것을 권고한다. 이렇게 OS레벨에서의 기술과 DBMS레벨에서의 기술이 서로 상충되면 문제를 해결하기 어렵기 때문이 아닐까 싶다
- **OS의 버퍼 캐시도 고려해야 한다.**
- OS에도 파일 캐시나 페이지 캐시라고 불리는 버퍼 캐시가 있다. 오라클의 버퍼 캐시와 같은 역할을 수행한다. 그렇기 때문에 오라클을 재기동하고 오라클의 버퍼 캐시를 비워도 OS가 재기동되지 않았다면 OS 버퍼 캐시에 데이터가 계속 남아 성능 측정에 오류가 생길 수 있다.(OS의 버퍼 캐시에서 데이터를 가져왔는데 디스크IO한 것이라고 판단할 수가 있다.)