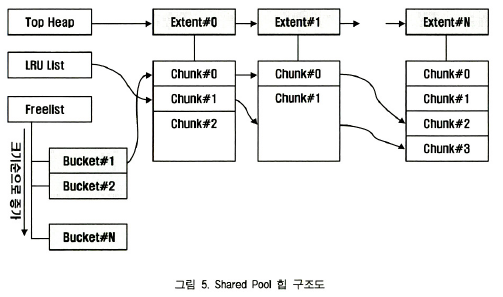
**Shared Pool / Library Cache와 OWI**

- Shared Pool과 힙

Shared Pool은 SGA의 Variable 영역에 속한다. Variable 영역은 Shared Pool + Java Pool + Large Pool + Streams Pool 등으로 이루어진다. Shared Pool은 다시 여러 종류의 메모리 영역으로 나누어진다. 대표적인 것들이 라이브러리 캐시와 로우 캐시이다. Shared Pool의 구성요소는 Permanent Area, Library Cache, Row Cache, Reserved Area로 구성된다.

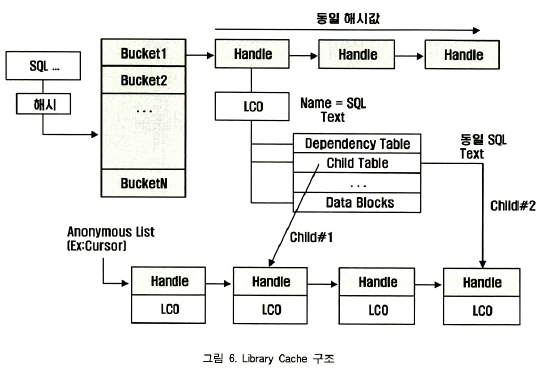
Shared Pool은 힙이라고 불리는 메모리 관리 기법을 이용해 관리된다. Shared Pool은 하나의 최상위 힙을 가지며, 최상위 힙은 다시 여러 개의 서브 힙을 포함한다. 힙은 여러 개의 익스텐트를 링크드 리스트 형식으로 거느린다. 하나의 익스텐트는 물리적으로 하나의 그래뉼을 사용한다. 하나의 익스텐트는 여러 개의 블록으로 구성된다. 서브힙의 경우에는 익스텐트의 크기는 가변적이다. 블록의 상태는 free, recr, freeabl, perm으로 나뉜다. free와 recr 상태인 블록들은 재사용이 가능하다. 힙은 프리리스트를 관리하는 버킷 테이블을 가진다. 각 버킷은 프리 블록에 대한 프리리스트를 링크드 리스트 형식으로 거느린다. 힙은 LRU 리스트를 가지고 있으며, 최상위 힙은 예약 프리리스트라는 별도의 프리리스트를 관리한다. 예약 프리리스트는 크기가 큰 객체를 저장하는 Shared Pool 내의 예약 영역에 대한 프리리스트 정보를 관리한다.

\* Shared Pool 힙으로부터 메모리를 할당 받고자 하는 모든 프로세스는 반드시 Shared Pool 래치를 획득해야 한다. 오라클은 9i부터 Shared Pool을 여러 개의 서브풀로 최대 7개까지 나누어서 관리할 수 있다. 이를 통해서 Shared Pool 래치 경합을 줄일 수 있다.



- 라이브러리 캐시 구조

라이브러리 캐시 영역은 SQL 문의 수행과 관련된 모든 정보를 관리하는 영역이다. 라이브러리 캐시 메모리는 Library Cache Manager(KGL)에 의해 관리되는데, KGL은 KGH를 이용해서 필요한 메모리 청크를 할당 받는다. 라이브러리 캐시 메모리는 해시 테이블 -> 버킷 -> 체인 -> 핸들 -> 오브젝트의 구조로 되어있다. 하나의 라이브러리 캐시 핸들은 하나의 라이브러리 캐시 오브젝트를 관리한다. 핸들은 실제 LCO에 대한 메타 정보 및 포인터 역할을 하며 LCO가 실제 정보를 가지고 있다.



LCO 정보 중 중요한 것은 크게 Dependency Table, Child Table(실제 SQL 커서 정보 저장하는 곳은 Child LCO), Data Blocks가 있다. 라이브러리 캐시 영역을 탐색하고자 하는 모든 프로세스는 반드시 해당 해시 버킷을 보호하는 라이브러리 캐시 래치를 획득해야 한다. 또한 라이브러리 캐시 락과 라이브러리 캐시 핀이라는 두 개의 락은 핸들과 LCO를 보호하는 역할을 한다.

- SQL 수행

1) 기본적인 문법체크와 권한체크 등을 수행한 후, 라이브러리 캐시 래치를 획득하고 라이브러리 캐시 영역에 동일한 LCO가 존재하는지 확인한다. 동일한 LCO가 존재하는 경우에는 8번 단계를 실행하게 되고, 이것을 소프트파싱이라고 한다.

2) 동일한 SQL 문장이 없다면, Shared Pool 래치를 획득하고 가장 적절한 크기의 프리 청크를 프리리스트에서 찾는다.

3) 만일 최적 크기의 프리 청크가 존재하지 않으면, 조금 더 큰 크기의 프리 청크를 찾아서 이를 쪼개어 사용한다. 쪼개고 남은 메모리 영역은 다시 적절한 프리리스트로 등록된다.

4) 모든 프리리스트를 탐색하고도 적절한 크기의 프리 청크를 찾지 못하면 LRU 리스트를 탐색한다.

5) LRU 리스트를 탐색하고도 적절한 크기의 청크를 확보하지 못하면 Shared Pool 내의 여유메모리 공간을 추가적으로 할당한다.

6) 위의 과정이 모두 실패하면 ORA-4031 에러가 발생한다.

7) 적절한 청크를 찾으면 SQL 문장에 해당하는 핸들에 대해 라이브러리 캐시 락을 Exclusive하게 획득하고 LCO정보를 생성한다. 생성되면 락을 Null모드로 변환하고, 라이브러리 캐시 핀을 Exclusive하게 획득한 후 실행 계획을 생성한다. 위 2~7번 과정을 하드파싱이라고 부른다.

8) 오라클은 SQL 커서에 대해 라이브러리 캐시 락과 핀을 Shared 모드로 획득하고 SQL 문장을 실행한다. 이 과정을 실행 단계라고 부른다. 기본적으로 SQL 커서와 동일한 모드로 라이브러리 캐시 락과 핀을 얻지만, DDL과 같은 경우는 Exclusive 모드로 획득한다.

9) 실행이 완료된 SQL 커서에 대해 데이터를 페치한다. 페치 단계에서는 SQL 커서에 대해 라이브러리 캐시 락을 Null모드로 변환하고 라이브러리 캐시 핀은 해제한다.

파싱 단계에서 사용된 CPU 시간과 파싱을 수행하는데 걸린 시간은 parse time cpu 통계값과 parse time elapsed 통계 값에 기록된다. 파싱을 수행하는 과정에서 래치나 락을 획득하기 위해 대기하는 시간이 길어지면 parse time elapsed 통계 값이 parse tie cpu 통계 값에 비해 매우 크게 나타날 수 있다. 또한 위 파싱 단계에서 과도한 하드파싱이 나타나면 Shared Pool 단편화가 발생하게 되고, ORA-4031 에러로 인해 실행이 중단된다.

오라클은 라이브러리 캐시 래치 경합을 줄이기 위해 여러 방안을 제시한다.

1) PL/SQL 블록 내에서 반복적으로 실행되는 SQL 커서에 대해서는 최초 실행시에만 파싱이 이루어지며, 이후에는 소프트파싱 없이 실행이 가능하다.

2) 세션 내부에 LCO의 위치를 캐싱하는 기능을 제공한다. 이것을 세션 커서 캐싱이라고 부른다. 이 기능을 사용할 경우 비록 소프트 파싱은 여전히 발생하고, 라이브러리 캐시 래치를 획득해야 하지만, 라이브러리 캐시 메모리 구조를 탐색하지 않고 바로 LCO의 위치를 찾아가게 되므로 라이브러리 캐시 래치를 보유하는 시간이 줄어든다. 그만큼 라이브러리 캐시 래치에 의한 경합도 줄어든다.