Posted by : Jaeeun Kim at Jun 12, 2019

Disk/File Manager

본 시리즈는 CUBRID의 Disk Manager와 File Manager의 내부구조와 동작과정을 소스코드레벨에서 분석하여 정리한 내용을 담는다. Disk/File Manager는 큐브리드가 Heap, Index등의 데이터를 OS의 파일에 담고 이들을 위해 공간을 할당, 조정한다. 이 글은 시리즈의 첫번째 포스팅으로 전체적인 아키텍처에 관한 개요와 용어들을 설명한다. 이어지는 포스팅에서는 개념, 아키텍쳐 뿐만 아니라 구조체와 함수단위에서 각 컴포넌트가 어떻게 동작하는지 살펴볼 것이다.

> 모든 내용은 CUBRID github 저장소의 버전은 10.2.0, develop branch: 7094ba33f61c을 기준으로 한다.

> 설명 중 함수명이나 변수명은 이텔릭체로 *var1, funtion()* 등으로 표기한다. 특히 함수의 경우 *function()*과 같이 인자없이 괄호를 추가하여 표기한다.

용어정리

- **볼륨 (Volume)**: 큐브리드에서 사용하는 OS가 제공하는 파일(open() 시스템콜을 통해 생성하는)을 말한다. 볼륨은 데이터 볼륨, 로그 볼륨, 백업 볼륨 등 여러 종류가 있지만 앞으로 언급하는 볼륨은 모두 Index와 Heap등이 담기는 데이터 볼륨(혹은 디스크 볼륨)을 가리키는 것으로 한다.
- **페이지 (Page)**: 페이지는 고정된 크기의 연속적인 데이터 블록(block)으로 큐브리드가 스토리지를 관리할 때 사용하는 가장 작은 단위이다. 큐브리드에는 로그페이지와 데이터페이지 두종류가 있지만 여기서는 데이터페이지만을 가리키는 것으로 한다. 이 페이지는 메모리로 로드되면, 버퍼매니저(Buffer Manager)에서 사용되는 Page Buffer와 매핑된다.
- **섹터 (Sector)**: 큐브리드가 스토리지를 다룰 때 사용하는 또 다른 단위로, 64개의 페이지들의 묶음을 이야기한다. 페이지 단위로 관리하는 것은 자원소모가 심하므로 묶음인 섹터로 일차적인 관리를 한다.
- **파일 (File)**: 큐브리드에서 이야기하는 파일이란, OS가 제공하는 파일이 아닌 특정 목적을 위해 예약된 섹터들의 묶음을 말한다. 파일은 섹터단위로 볼륨의 공간을 예약하고, 필요에 따라 섹터내의 페이지를 할당하여 사용한다. 각각의 파일은 하나의 테이블, 하나의 인덱스, 나중에 이야기할 파일 트래커(File Tracker)등의 정보를 담는다.
- **섹터 예약 (Reservation)**: 볼륨의 섹터를 사용하기로 하는 행위를 말한다. 반대로 사용을 중지하고 반납하는 행위를 섹터 예약 해제 (Unreservation)이라고 한다.
- **페이지 할당 (Allocation)**: 파일에서 예약한 섹터중 한 페이지를 사용하기로 하는 행위를 말한다. 반대로 사용을 중지하고 반납하는 행위를 페이지 할당해제(Deallocation)라고 한다.

> 앞으로 파일은 상기에 서술된 큐브리드의 파일을 이야기하고, OS의 파일을 이야기할 때는 OS파일 혹은 볼륨으로 표현한다.

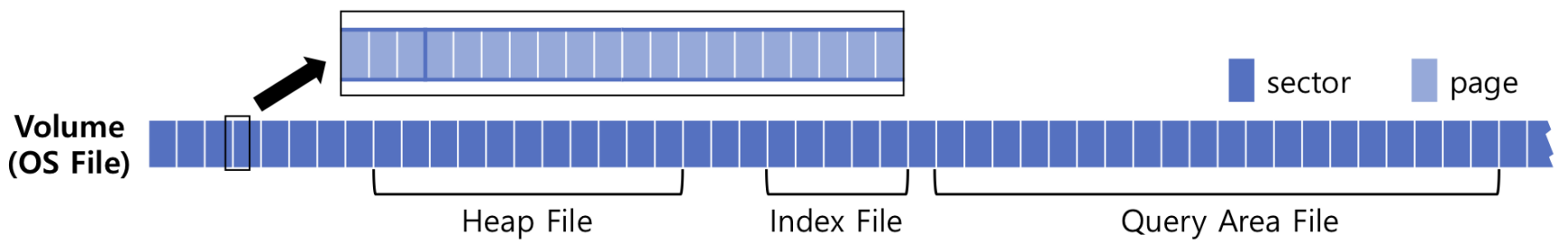

Figure 1: Volume Overview

디스크 매니저와 파일매니저

디스크 매니저: 볼륨공간 전체를 관리하며 섹터들의 예약여부를 트래킹한다. 섹터들의 예약관리가 주역할이며, 모든 섹터가 다 예약돼었을 경우 볼륨의 크기(OS파일의 크기)를 늘리거나 볼륨의 수를 늘림으로서 추가적인 섹터를 확보한다. 디스크매니저 관련함수들은 *disk_** prefix로 시작한다.

파일매니저: 큐브리드의 내부파일들을 관리하며 디스크매니저로부터 섹터들을 할당받고 파일 내에서의 페이지할당여부를 트래킹한다. 페이지들의 할당 관리가 주 역할이며, 모든 페이지를 할당하여 추가적인 공간이 필요할 경우 디스크 매니저에게 추가적인 섹터를 요청한다. 파일매니저 관련 함수들은 *file_** prefix로 시작한다.

> 앞으로 설명할 볼륨관련 데이터나 함수는 디스크 매니저, 파일관련 데이터나 함수는 파일매니저라고 보면 된다.

볼륨 (Volume)

데이터볼륨은 영구(Permanent)볼륨과 임시(Temporary)볼륨으로 나뉜다.

- **영구볼륨**: 테이블, 인덱스, 시스템데이터 등이 담기는 볼륨으로 한번 만들어지면 영구히 존재하는 볼륨을 말한다.
- **임시볼륨**: 정렬(sorting)과정이나 쿼리결과를 가져오는 과정에서의 임시데이터들을 담는 볼륨을 말하며, 데이터베이스 종료와 재시작시 모두 삭제된다.

그렇다고해서 영구볼륨의 데이터는 항상 보존되어야 하는 데이터만(즉, 리커버리의 대상이되는)이 담기는 것은 아니고 addvoldb로 임시목적(purpose)의 영구볼륨을 만들경우 임시데이터가 담길 수도 있다. 볼륨에 대한 개념적인 내용은 큐브리드 메뉴얼을 참고하자.

볼륨은 언제만들어질까?

이해를 돕기위하여 볼륨의 생성이 언제 일어나는지를 알아보자. 영구볼륨이 생성되는 경우는 다음과 같다.

1. 데이터베이스를 createdb명령을 통하여 생성할 때
2. 사용중인 영구볼륨들이 모두 가득차서 추가적인 공간이 필요할 때
3. addvoldb를 통하여 사용자가 직접 볼륨을 추가할 때

데이터가 지속적(durable)으로 보관되기 위해서는 영구볼륨이 필요하다. 지속성은 데이터베이스의 기본 속성이므로 데이터베이스가 처음 생성될 때 기본적으로 영구볼륨이 생성된다. 생성된 영구볼륨을 사용하는 과정에서 데이터가 가득차게 되면 새로운 볼륨이 추가로 생성되는데, 이 때는 시스템파라미터에 있는 정보를 바탕으로 최소한의 크기로 볼륨이 생성되고, 데이터가 채워짐에 따라서 볼륨의 크기가 늘어난다. 그러다가 다시 볼륨의 최대크기까지 데이터가 입력되면 새로운 볼륨이 생성되어 이를 처리한다.

> 영구볼륨의 제거나 축소는 없고, deletedb를 통해서 전체 데이터베이스를 제거할때만 볼륨들이 모두함께 제거되는 것으로 보인다.

그렇다면 임시볼륨은?

쿼리중간결과등의 임시데이터는 임시볼륨에만 담길 수 있는 것은 아니다. 임시목적의 영구볼륨이 존재할 경우에는 기본적으로 이 영구볼륨에 담기게 되며 공간이 부족해질 경우에 임시볼륨이 만들어져 데이터를 저장한다. 이렇게 생성된 임시볼륨은 이후에 데이터베이스가 종료되거나 재시작될 때 모두 제거된다.

> 만약 이미 임시 볼륨이 있다면?
>
> 임시데이터를 위한 섹터를 할당하려할 때 이미 이전에 만들어진 임시볼륨이 존재한다고 해도, 먼저 임시목적의 영구볼륨에서 섹터를 할당시도한다.

파일 (File)

앞서 이야기했듯이 큐브리드에서 말하는 파일은 OS에서 제공하는(open등의 시스템콜로 생성하는) OS file이 아니라, 큐브리드만의 독자적인 유닛으로 볼륨내의에서 **하나의 목적으로 할당된 섹터들의 논리적인 집합**이다. 파일을 생성하면 볼륨의 섹터들을 할당받고 파일내에서는 이를 가장 작은 단위인 페이지로 나누어서 관리하며, 할당받았던 섹터의 페이지를 모두 사용하면 추가적인 섹터들을 추가적으로 할당 받는다. 파일도 볼륨과 같이 영구파일과 임시파일 두가지로 분류할 수 있다.

- **영구파일**: 각각의 파일은 인덱스, 힙데이터, 파일트래커 등의 특정한 목적을 가지고 있으며, 변경이 일어나면 이는 로그로 기록되어 리커버리의 대상이 된다. 영속적으로 보관할 데이터이므로 업데이트 비용이 비싸며 관리 오버헤드가 있다.
- **임시파일**: 쿼리나 정렬의 중간결과들이 일시적으로 쓰여지는 파일로 기본적으로 임시파일을 사용하는 트랜잭션이 종료되면 제거된다. 혹은, 필요에 따라 트랜잭션종속에서 벗어나 쿼리매니저에서 관리되기도 한다. 임시파일은 사용하는 순간에만 유효하면 되므로 영구파일에 비해 연산자체도 단순하고 관리 오버헤드도 적다.

파일은 위의 두가지 분류 뿐만 아니라 파일의 목적에 따라 heap, btree, catalog등의 파일타입으로도 나눌 수 있다.

Numerable 속성

파일은 Numerable 속성을 지닐 수 있다. 기본적으로 파일에 할당되는 페이지는 순서가 없다. 물리적으로 연속적인 페이지만을 할당받는 것은 아니며 여러 섹터에 흩어져 있는 페이지들을 할당받는다. 심지어 할당받은 각 페이지가 속한 섹터들도 연속적이지 않을 수 있으며 여러볼륨에 흩어져 있을 수도 있다. 이러한 페이지들에 논리적인 순서를 부여하는 속성을 numerable이라 한다. 이 속성을 지니면 파일에 할당된 페이지(정확히는 유저페이지만)들을 할당된 순서대로 인덱스를 통하여 접근할 수 있다. 이는 extensible hash나 external sorting의 경우에 유용하게 사용될 수 있다.

> 이 중 extensible hash는 현재 deprecated되어 있고 호환성을 위해서만 남아있는 것으로 보인다.

이번 글에서는 디스크매니저와 파일매니저에서 쓰이는 개념들에 대하여 정리하였다. 앞으로의 글에서는 이를 바탕으로 어떤 방식으로 파일과 볼륨이 관리되는지 자세히 알아본다. 이후의 내용은 다음과 같다.

- 볼륨은 어떻게 관리될까?
- 섹터 예약은 어떻게 이루어질까?
- 섹터 예약시 볼륨에 섹터가 부족하면?
- 큐브리드 파일은 어떻게 관리될까?
- 페이지 할당은 어떻게 이루어질까?
- 파일의 생성과 제거