큐브리드 파일 아키텍쳐
(CUBRID File Architecture)

Posted by : Jaeeun Kim at Jul 16, 2019

큐브리드 파일은 어떻게 관리될까?

- CUBRID File Architecture -

이번 글부터는 파일과 파일매니저에 대하여 알아본다. 파일매니저는 볼륨매니저로부터 섹터를 예약하여 파일을 생성/제거하고, 페이지를 할당하고, 필요할 경우 추가적인 섹터를 요청하는 등 파일의 공간을 관리하는 역할을 한다. 파일도 볼륨과 마찬가지로 저장되는 데이터의 목적에 따라 영구파일과 임시파일로 나뉜다. 영구파일과 임시파일은 전체적인 구조는 같으나 파일테이블의 이용방법이나 연산들의 동작이 조금씩 다르다. 이와 함께 파일의 numerable속성을 알아보고 마지막으로 파일헤더를 보면서 파일이 구체적으로 어떠한 데이터를 포함하는지 알아본다.

이 포스트에서는 다음의 내용을 다룬다.

- File Overview
  - 볼륨과 파일
  - 영구파일과 임시파일
  - 파일내 페이지의 종류
  - Numerable 속성
- 파일 헤더 (File Header)

File Overview

볼륨과 파일

파일은 특정 목적을 위한 섹터들의 묶음으로 큐브리드에서 데이터를 관리하는 핵심적인 단위이다. 볼륨과 파일 둘다 최소한의 IO단위인 페이지들의 집합이지만 볼륨은 물리적인 묶음(OS 파일)이고 파일은 논리적인 묶음이다. 파일은 볼륨매니저로부터 예약한 섹터들로 이루어져 있으며, 이 섹터들은 연속적이지 않을 수 있고 심지어는 여러 볼륨에 걸쳐있을 수도 있다. 파일은 섹터단위로 페이지들을 확보하고 필요한 만큼 할당하여 사용하고 예약해둔 섹터를 모두사용하면 추가적인 섹터를 예약한다. 앞서 살펴보았던 볼륨매니저도 결국 이 파일들의 공간할당 요청을 처리하기 위해 섹터들을 관리하고 OS로부터 추가적인 공간을 요청하기도 하는 것이다.

아래의 그림을 보면 파일과 볼륨, 파일매니저와 볼륨매니저의 관계를 알 수 있다. 여기서 파일은 별도의 물리적 개체가 아니다. assign한다고 표현하였지만 물리적으로는 페이지들(섹터들)을 사용중이라고 표시만 할 뿐이다. 그림에서 사각형은 페이지라고 생각해도 되고 페이지라고 생각해도 된다. 실제로는 페이지들의 묶음인 섹터단위로 예약한다. 예약한 섹터는 파일이 오퍼레이션을 수행하면서 페이지단위로 할당해서 사용한다.

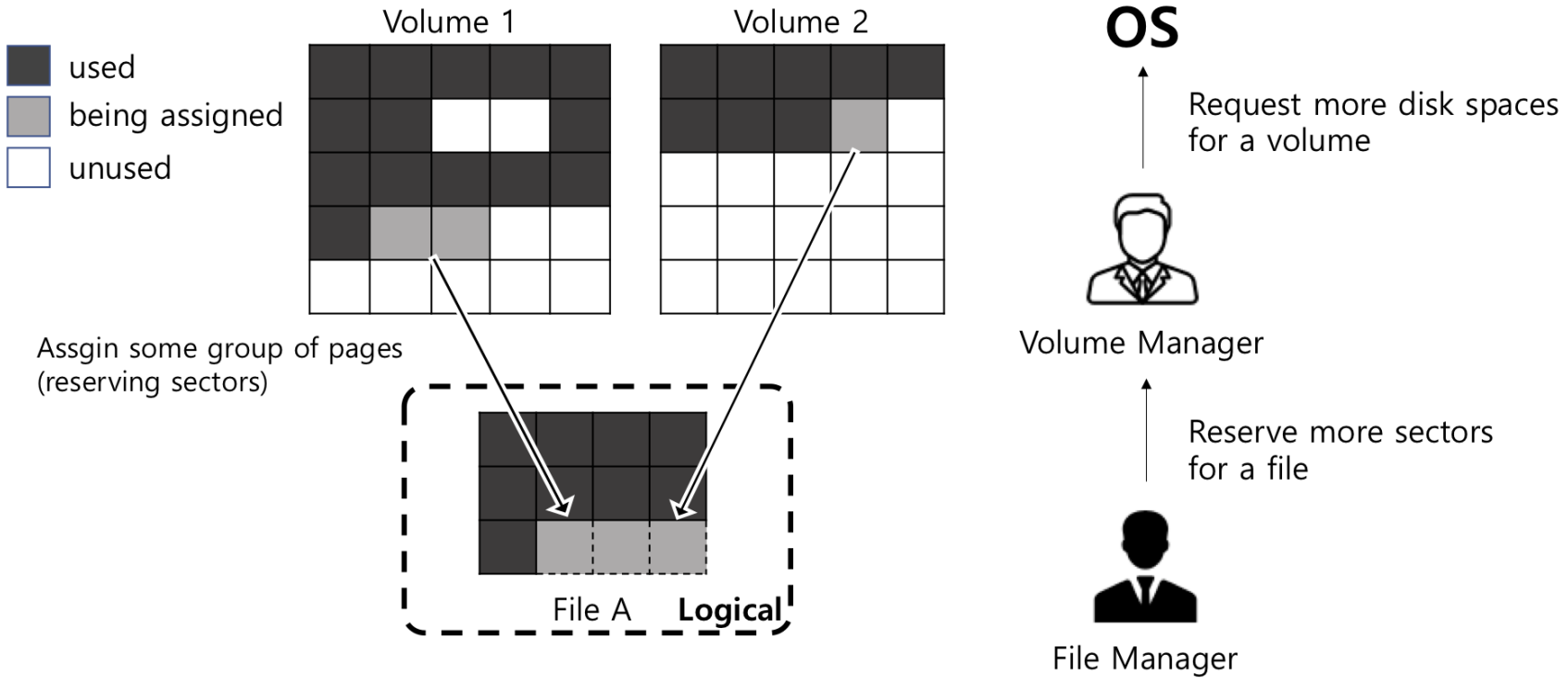

Figure 1: 파일과 볼륨

영구파일과 임시파일

영구파일과 임시파일의 구분은 볼륨과 같다. 영구파일은 테이블과 인덱스와 같이 영구적으로 데어터를 저장할 목적으로 사용되고 트랜잭션이 진행되면서 임시파일은 쿼리결과나 정렬결과등 일시적으로 데이터를 저장하기 위하여 사용한다. 임시파일의 경우엔 기본적으로 트랜잭션이 commit되거나 abort될 때 파괴된다. 영구파일은 영구목적으로 섹터를 예약하고, 임시파일은 임시목적으로 섹터를 예약한다. 각 파일이 어떤식으로 섹터를 예약하는지는 섹터예약에 관한 글을 참조하자.

볼륨내에 어떤 파일들이 있는지 관리되어야 하기 때문에 영구파일의 경우는 파일트래커(File Tracker)는 특수한 파일을 통해 추적된다. 반면에 임시파일의 경우에는 트랜잭션이 끝나고 참조되지 않으므로 따로 추적되지 않는다. 대신에 임시파일같은 경우는 매 트랜잭션마다 생성과 파괴를 반복할 것이므로 이를 위해 캐싱(Temp Cache)을 한다. 또 임시파일은 곧 사라질 파일이기 때문에 공간관리가 단순화 될 수 있다. 파일트래커와 임시파일의 연산은 이후에 자세히 살펴보도록 한다.

Numerable 속성

파일은 numerable이라는 속석을 가질 수 있다. 이 속성은 파일내에서 할당된 유저페이지들이 순서를 가진다. 기본적으로 파일은 페이지의 할당여부만을 관리하고 각 페이지들의 관계나 순서등은 관리하지 않는다. 페이지들이 어떤 관계를 가지기 위해서는 각 파일이 내부적으로 페이지들을 링크로 연결하는 등 관계를 형성해야 한다. 하지만 만약 numerable 속성을 주게되면 이를 위한 파일 테이블을 추가하고 새로운 페이지를 할당할 때마다 이를 저장하여 *file_numerable_find_nth()* 인터페이스를 통하여 순서대로 접근할 수 있도록 한다.

Numerable 파일의 용도

*file_numerable_find_nth()*의 호출자들을 살펴보면 ehash와 external sort과정에서 사용하는 것을 볼 수 있다. ehash의 경우 현재 deprecated된 것으로 보이므로 정렬과정에서만 사용되는 듯하다.

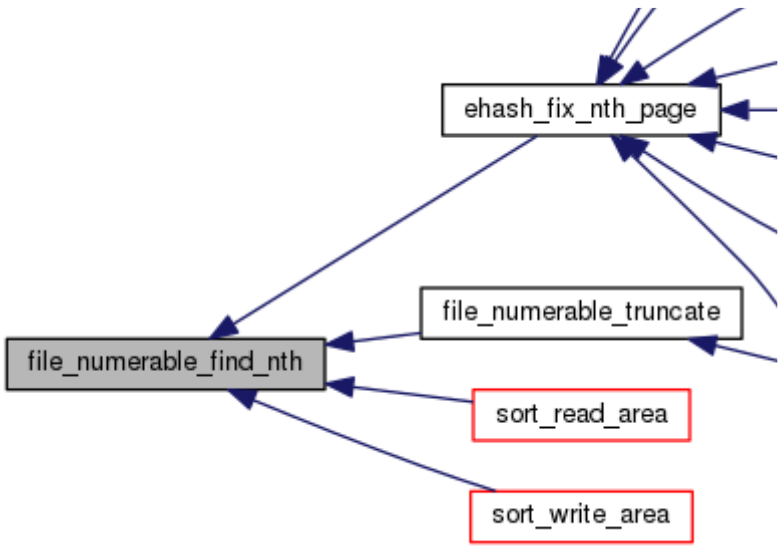

Figure 1: file_numerable_find_nth 호출자

파일테이블 페이지와 유저 페이지

볼륨매니저가 OS로부터 공간을 할당받고 할당받은 공간을 섹터단위로 나누어 섹터의 예약요청을 처리한 것과 같이, 파일매니저는 볼륨으로부터 공간(섹터)을 할당 받고 할당받은 섹터를 페이지단위로 나누어 파일내에서 페이지의 할당을 처리한다. 이를 위해 볼륨매니저는 섹터관리를 위한 시스템페이지와 일반 페이지로 페이지들을 구분하였고, 시스템페이지는 볼륨헤더페이지(*PAGE_VOLHEADER*)와 섹터테이블 페이지(*PAGE_VOLBITMAP*)로 구분되었다. 이와 유사하게 파일또한 다음과 같이 페이지들로 구분된다.

- **파일 테이블 페이지 (*PAGE_FTAB*)**: 파일내의 페이지 관리를 위한 정보를 담고 있는 페이지
  - **파일 헤더 페이지**: 파일의 예약섹터정보, 페이지할당정보 등을 지니고 있다. 파일 헤더 페이지 또한 파일에 대한 메타데이터와 함께 파일 테이블들을 지니고 있고 페이지 타입이 *PAGE_FTAB*이다.
  - **파일 테이블 페이지**: 파일 내의 페이지들의 할당정보를 담고 있는 페이지. 파일 테이블은 볼륨으로부터 할당받은 섹터마다 섹터의 어떤 페이지들이 할당되어 사용되고 있는지를 추적한다. 테이블 페이지의 종류에 따라 페이지 할당 비트맵을 포함하지 않을수도 있다.
- **유저 페이지 (PAGE_*)**: 페이지 테이블 페이지를 제외한 페이지들을 말한다. 파일의 용도에 따라 여러가지 페이지 타입이 될 수 있다.

> 파일 헤더페이지가 반드시 파일의 첫번째 페이지인 것은 아니다. 이는 Vacuum의 dropped file과 관련된 것으로 보인다. 추후 MVCC, VACUUM 관련글에서 다뤄보도록 하겠다.

페이지의 타입 (참고용)

```
typedef enum
{
PAGE_UNKNOWN = 0,     /* used for initialized page */
PAGE_FTAB,            /* file allocset table page */
PAGE_HEAP,            /* heap page */
PAGE_VOLHEADER,       /* volume header page */
PAGE_VOLBITMAP,       /* volume bitmap page */
PAGE_QRESULT,         /* query result page */
PAGE_EHASH,           /* ehash bucket/dir page */
PAGE_OVERFLOW,        /* overflow page (with ovf_keyval) */
PAGE_AREA,            /* area page */
PAGE_CATALOG,         /* catalog page */
PAGE_BTREE,           /* b+tree index page (with ovf_OIDs) */
PAGE_LOG,         /* NONE - log page (unused) */
PAGE_DROPPED_FILES,       /* Dropped files page.  */
PAGE_VACUUM_DATA,     /* Vacuum data. */
PAGE_LAST = PAGE_VACUUM_DATA
} PAGE_TYPE;
```

파일 헤더 (*FILE_HEADER*)

각 파일마다 기본적으로 한개의 파일헤더페이지를 지니고, 파일헤더페이지의 첫 부분에는 파일헤더(*FILE_HEADER*)가 들어간다. 파일 헤더에는 파일에 대한 기본적인 정보(파일타입과 무관하게 공통적으로 가지는 정보들), 파일이 예약한 섹터들, 파일 내에서 할당된 페이지들을 관리하기 위한 정보들이 들어가 있다. 또한 numerable, temp file의 연산을 위한 캐싱변수들이 포함된다. 이를 정리해보면 다음과 같다.

FILE_HEADER

타입	변수	설명
파일 정보	INT64 time_creation	파일이 만들어진 시간
	VFID self	자기식별자 (fileid, volid)
	FILE_TABLESPACE tablespace	파일확장시 최소, 최댓값 등이 담김 볼륨과는 다르게 총 확장가능한 최댓값이 아니라, 한 확장연산시 확장량을 결정한다.
	FILE_DESCRIPTORS descriptor	파일타입에 따른 논리적인 식별자
	FILE_TYPE type	파일 타입
	INT32 file_flags	파일의 속성, 현재는 임시파일인지와 numerable인지의 여부가 담김. FILE_FLAG_NUMERABLE/TEMPORARY
	VPID vpid_sticky_first	파일의 sticky page로 페이지할당해제의 대상이 되지 않는다.
	VOLID volid_last_expand	파일 확장시 가장 마지막에 섹터를 예약한 볼륨. 섹터예약시 어떤 볼륨을 먼저 확인할지 힌트로 사용되기 위한 용도로 보이나, 현재는 사용되고 있지 않다.
	INT16 offset_to_partial_ftab	파일헤더페이지내에서 partial ftab의 시작위치
	INT16 offset_to_full_ftab	파일헤더페이지내에서 full ftab의 시작위치
	INT16 offset_to_user_page_ftab	파일헤더페이지내에서 user page ftab의 시작위치
페이지 카운트	int n_page_total	파일의 총 페이지 수

	int n_page_user	파일의 유저페이지 수
	int n_page_ftab	파일의 ftab 페이지 수
	int n_page_free	파일의 할당되지 않은 페이지 수
섹터 카운트	int n_sector_total	예약한 총 섹터 수
	int n_sector_partial	예약한 섹터 중 일부 페이지가 할당에 사용된 섹터 수
	int n_sector_full	예약한 섹터 중 모든 페이지가 할당에 사용된 섹터 수
	int n_sector_empty	예약한 섹터 중 어떤 페이지도 할당에 사용되지 않은 섹터 수 empty sector도 partial sector이다. n_sector_empty <= n_sector_partial
임시파일을 위한 캐싱 변수	VPID vpid_last_temp_alloc	임시파일의 페이지 할당 시 마지막 페이지를 할당받은 partial table의 페이지 ID와 페이지 내부의 offset. 임시파일의 페이지를 할당 받는 위치. 이 후 임시파일을 다룰 때 자세히 다루겠다.
	int offset_to_last_temp_alloc	
Numerable 파일을 위한 캐싱 변수	VPID vpid_last_user_page_ftab	마지막으로 할당된 user page에 대한 테이블 엔트리가 추가된 user page ftab 페이지의 ID. 유저페이지 추가시 테이블엔트리를 추가할 위치.
	VPID vpid_find_nth_last	Numerable파일의 주요 사용처인 external sort를 위한 캐싱변수. 유저페이지에 대한 마지막 순차접근의 위치를 저장한다.
	int first_index_find_nth_last	
예약 변수	INT32 reserved0/1/2/3	예약변수

Sticky Page

보통 파일이 만들어질 때, 파일헤더페이지를 제외하고 각 파일타입마다 추가적인 페이지를 할당받고 이를 sticky page로 등록한다. 페이지 할당이 순차적이지 않으므로 파일헤더가 아닌 각 파일타입에 맞는 정보가 담기거나 파일타입에 맞게 파일내의 페이지를 탐색할 수 있는 방법이 필요하며, 이를 sticky page를 통해 해결하는 것으로 보인다.

식별자 정리 : VSID, VPID, VFID

타입	섹터	페이지	파일
int32_t	sectid	pageid	fileid
short	volid	volid	volid

volid, sectid, pageid는 각각 순차적으로 부여되는 식별자이다. fileid는 파일이 생성된 페이지(파일헤더페이지)의 pageid를 식별자로써 사용한다. 이를통해 sectid, pageid, fileid를 이용하면 각 개체식별 뿐만 아니라 바로 물리적인 위치를 찾아갈 수도 있다.

이어서 다룰 파일매니저 내용은 다음과 같다.

1. 페이지 할당/해제, 파일확장
2. 파일의 생성과 파괴 (with File Tracker and File Temp Cache)