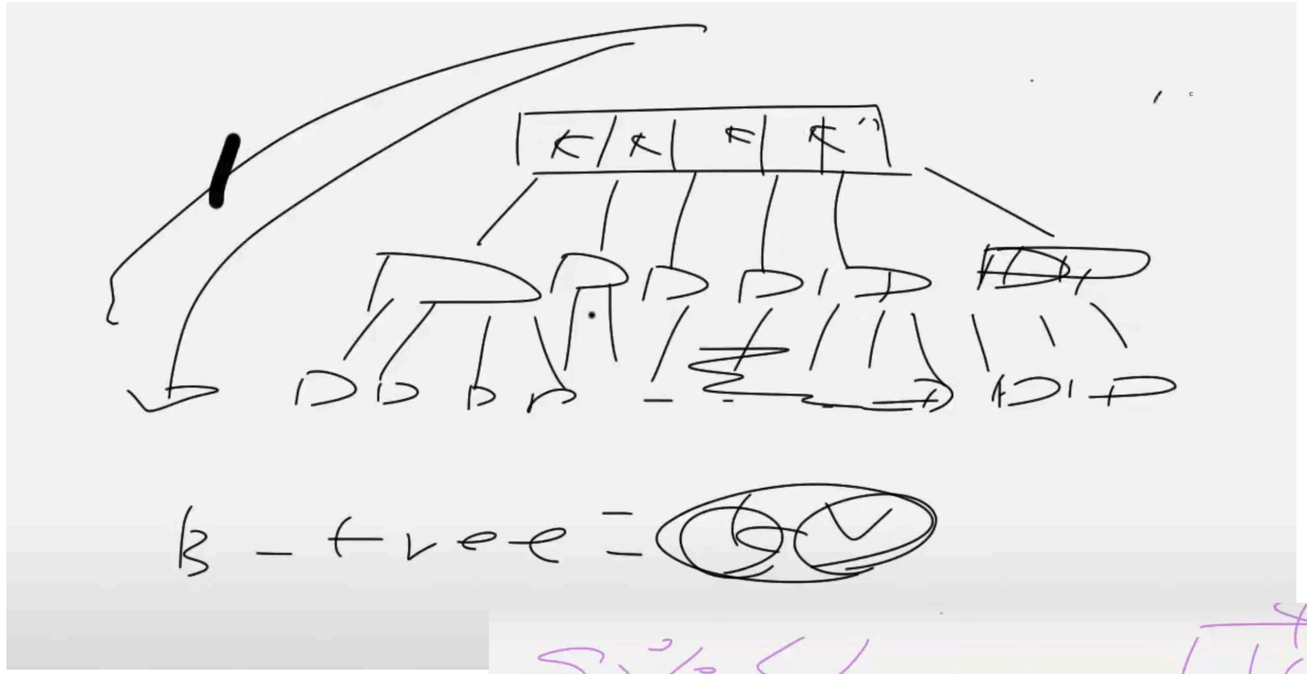
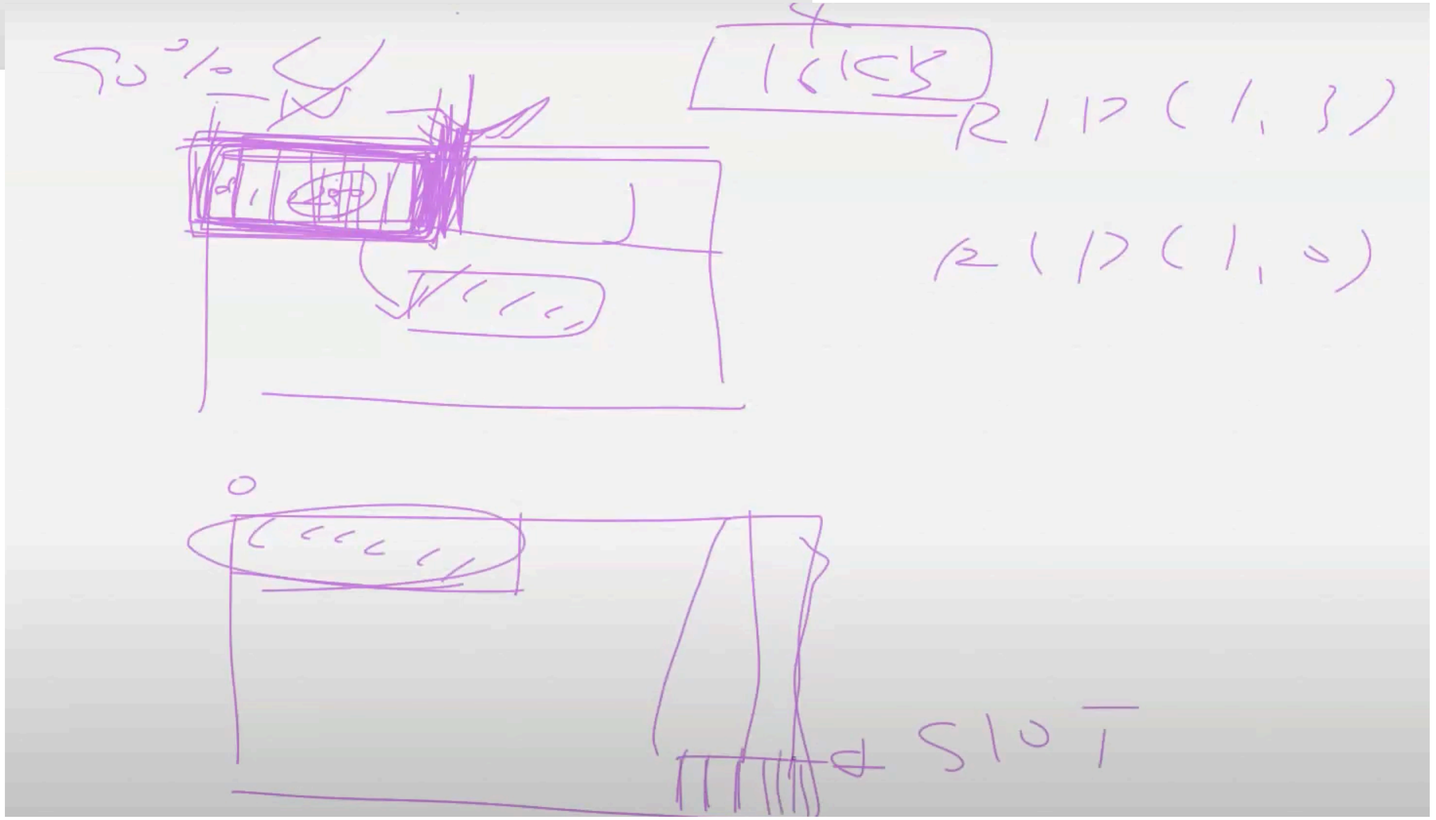
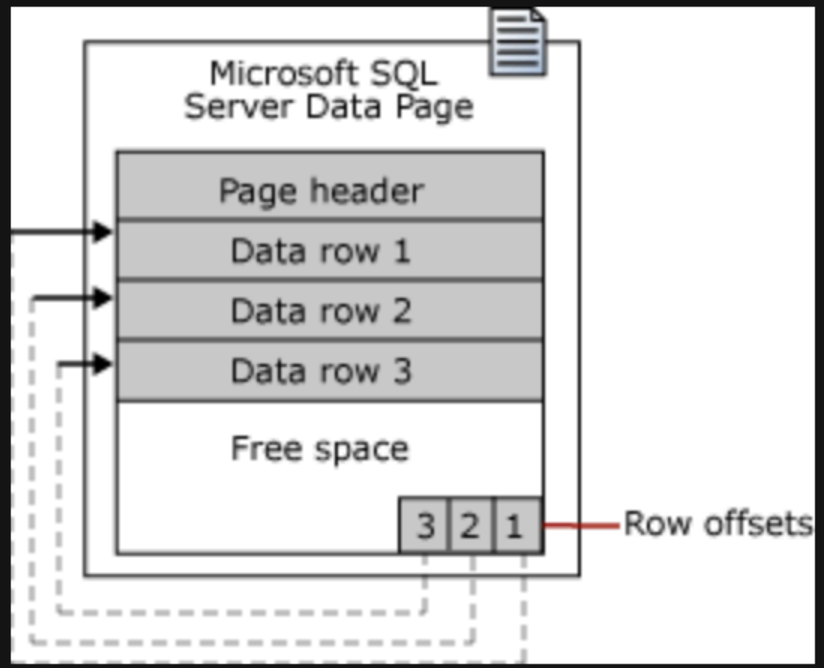




〈출처: honux〉

데이터 행은 머리글 바로 다음부터 시작하여 페이지에 차례로 나옵니다. 행 오프셋 테이블은 페이지 끝에서 시작하는데 각 행 오프셋 테이블에는 해당 페이지에 있는 각 행에 대한 항목이 하나씩 있습니다. 각 행 오프셋 항목은 행의 첫 번째 바이트가 페이지 시작 지점에서 얼마나 떨어져 있는지를 기록합니다. 따라서 행 오프셋 테이블 함수는 SQL Server가 페이지에서 행을 빠르게 찾는 데 도움이 됩니다. 행 오프셋 테이블의 항목 순서는 페이지의 행 순서의 역순입니다.



〈출처:https://docs.microsoft.com/ko-kr/sql/relational-databases/pages-and-extents-architecture-guide?view=sql-server-ver15〉

Google

InnoDB Page Structure

코드스쿼드 2022 BE 로니

학습 자료

https://dev.mysql.com/doc/internals/en/innodb-page-structure.html

https://dev.mysql.com/doc/internals/en/innodb-record-structure.html

https://www.codetd.com/en/article/14007258

! 자세한 설명은 codetd article에 있으나.
위 article에 작성자 및 정보 출처가 표시되어 있지않아서 100% 맞는 내용이라고 판단하기는 어렵습니다. !

! 제가 이해하기로는 공식 문서와 codetd article의 설명이 다소 차이가 있는 듯 했습니다. !

22.2.1.6 Page Directory

The Page Directory part of a page has a variable number of record pointers. Sometimes the record pointers are called "slots" or "directory slots". Unlike other DBMSs, `InnoDB` does not have a slot for every record in the page. Instead it keeps a sparse directory. In a fullish page, there will be one slot for every six records.

The slots track the records' logical order (the order by key rather than the order by placement on the heap). Therefore, if the records are `'A''B''F''D'` the slots will be `(pointer to 'A') (pointer to 'B') (pointer to 'D') (pointer to 'F')`. Because the slots are in key order, and each slot has a fixed size, it's easy to do a binary search of the records on the page via the slots.

(Since the Page Directory does not have a slot for every record, binary search can only give a rough position and then `InnoDB` must follow the "next" record pointers. `InnoDB`'s "sparse slots" policy also accounts for the n_owned field in the Extra Bytes part of a record: n_owned indicates how many more records must be gone through because they don't have their own slots.)

〈출처:https://dev.mysql.com/doc/internals/en/innodb-page-directory.html〉

다른 DBMS들과는 다르게 페이지 내의 모든 레코드가 각각 하나의 slot를 갖지는 않는다.

"sparse slots" policy

n_owned field

InnoDB Record Structure

- The FIELD START OFFSETS : 다음 필드 시작 위치들의 목록, 역순
  - 컬럼1, 2, 3이 있고 그 length가 각각 1, 2, 4라면 07 03 01
- The EXTRA BYTES : 레코드를 읽고 저장할 때 필요한 정보(n_owned 변수가 있는 부분)
- The FIELD CONTENTS : 실제 데이터가 저장된 부분

```
19 17 15 13 0C 06 Field Start Offsets /* First Row */
00 00 78 0D 02 BF Extra Bytes
00 00 00 00 04 21 System Column #1
00 00 00 00 09 2A System Column #2
80 00 00 00 2D 00 84 System Column #3
50 50 Field1 ‘PP'
50 50 Field2 'PP'
50 50 Field3 'PP'
```

InnoDB Record Structure

The EXTRA BYTES

n_owned

이 레코드가 가지고 있는 레코드들의 수 ?

next

다음 레코드를 가리키는 포인터

(singly linked list 형태)

Name	Size	Description
info_bits:	??	??
()	1 bit	unused or unknown
()	1 bit	unused or unknown
deleted_flag	1 bit	1 if record is deleted
min_rec_flag	1 bit	1 if record is predefined minimum record
n_owned	4 bits	number of records owned by this record
heap_no	13 bits	record's order number in heap of index page
n_fields	10 bits	number of fields in this record, 1 to 1023
1byte_offs_flag	1 bit	
next 16 bits	16 bits	Pointer to next record in page
TOTAL	48 bits	??

InnoDB Page Structure

Leaf node에 해당하는 data page기준으로 설명

page는 7파트로 구분되어 있다.

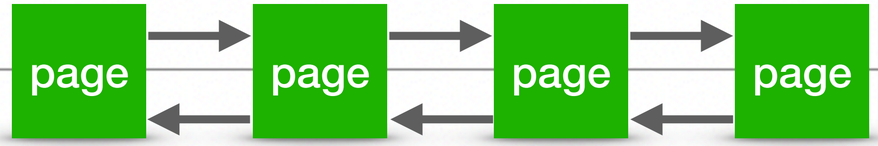
(InnoDB에서는 page를 block이라고도 부름)

InnoDB Page Structure

Fil Header

Name	Size	Remarks
FIL_PAGE_SPACE	4	4 ID of the space the page is in
FIL_PAGE_OFFSET	4	ordinal page number from start of space
FIL_PAGE_PREV	4	offset of previous page in key order
FIL_PAGE_NEXT	4	offset of next page in key order
FIL_PAGE_LSN	8	log serial number of page's latest log record
FIL_PAGE_TYPE	2	current defined types are: FIL_PAGE_INDEX, FIL_PAGE_UNDO_LOG, FIL_PAGE_INODE, FIL_PAGE_IBUF_FREE_LIST
FIL_PAGE_FILE_FLUSH_LSN	8	"the file has been flushed to disk at least up to this lsn" (log serial number), valid only on the first page of the file
FIL_PAGE_ARCH_LOG_NO	4	the latest archived log file number at the time that FIL_PAGE_FILE_FLUSH_LSN was written (in the log)

이전, 이후 페이지를 가리키는 포인터
(Doubly Linked List 형태)



〈출처:https://dev.mysql.com/doc/internals/en/innodb-fil-header.html〉

InnoDB Page Structure

Page Header

Name	Size	Remarks
PAGE_N_DIR_SLOTS	2	number of directory slots in the Page Directory part; initial value = 2
PAGE_HEAP_TOP	2	record pointer to first record in heap
PAGE_N_HEAP	2	number of heap records; initial value = 2
PAGE_FREE	2	record pointer to first free record
PAGE_GARBAGE	2	"number of bytes in deleted records"
PAGE_LAST_INSERT	2	record pointer to the last inserted record
PAGE_DIRECTION	2	either PAGE_LEFT, PAGE_RIGHT, or PAGE_NO_DIRECTION
PAGE_N_DIRECTION	2	number of consecutive inserts in the same direction, for example, "last 5 were all to the left"
PAGE_N_RECS	2	number of user records
PAGE_MAX_TRX_ID	8	the highest ID of a transaction which might have changed a record on the page (only set for secondary indexes)
PAGE_LEVEL	2	level within the index (0 for a leaf page)
PAGE_INDEX_ID	8	identifier of the index the page belongs to
PAGE_BTR_SEG_LEAF	10	"file segment header for the leaf pages in a B-tree" (this is irrelevant here)
PAGE_BTR_SEG_TOP	10	"file segment header for the non-leaf pages in a B-tree" (this is irrelevant here)

〈출처:https://dev.mysql.com/doc/internals/en/innodb-fil-header.html〉

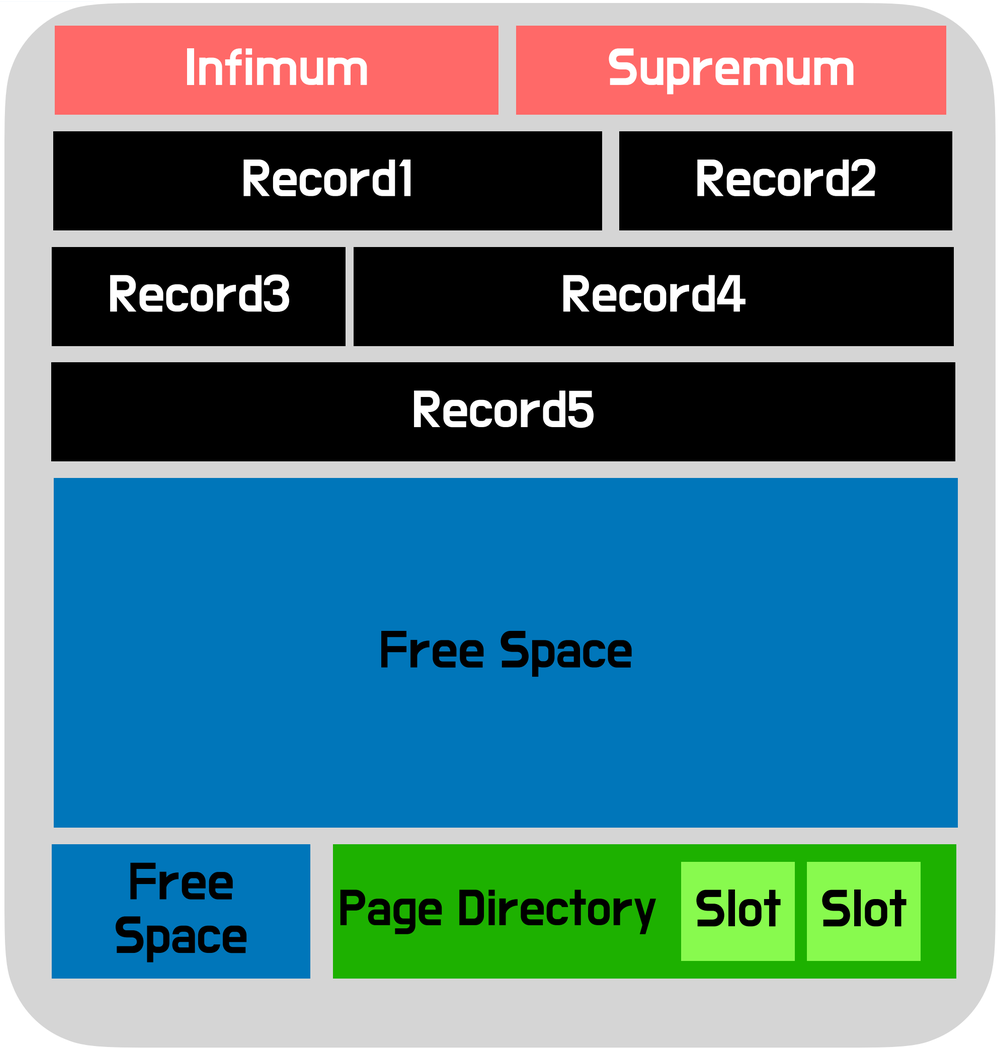
Infimum + Supremum Records

- "Infimum" 및 "Supremum"은 정렬된 집합의 외부 경계를 나타내는 수학 용어
- An infimum is Greatest Lower Bound (GLB)
- A supremum is the Least Upper Bound (LUB)
- 레코드를 서칭할 때 'get-prev', 'get-next' 시에 범위를 넘어가지 않도록 하는 barrier 역할
- User Record들은 Infimum과 Supremum 사이에 있음

Initially, they both exist on the root page, but as the index grows,
the infimum record will exist on the first or lowest leaf page
and the supremum will the last record on the last or greatest key page.

InnoDB Page Structure

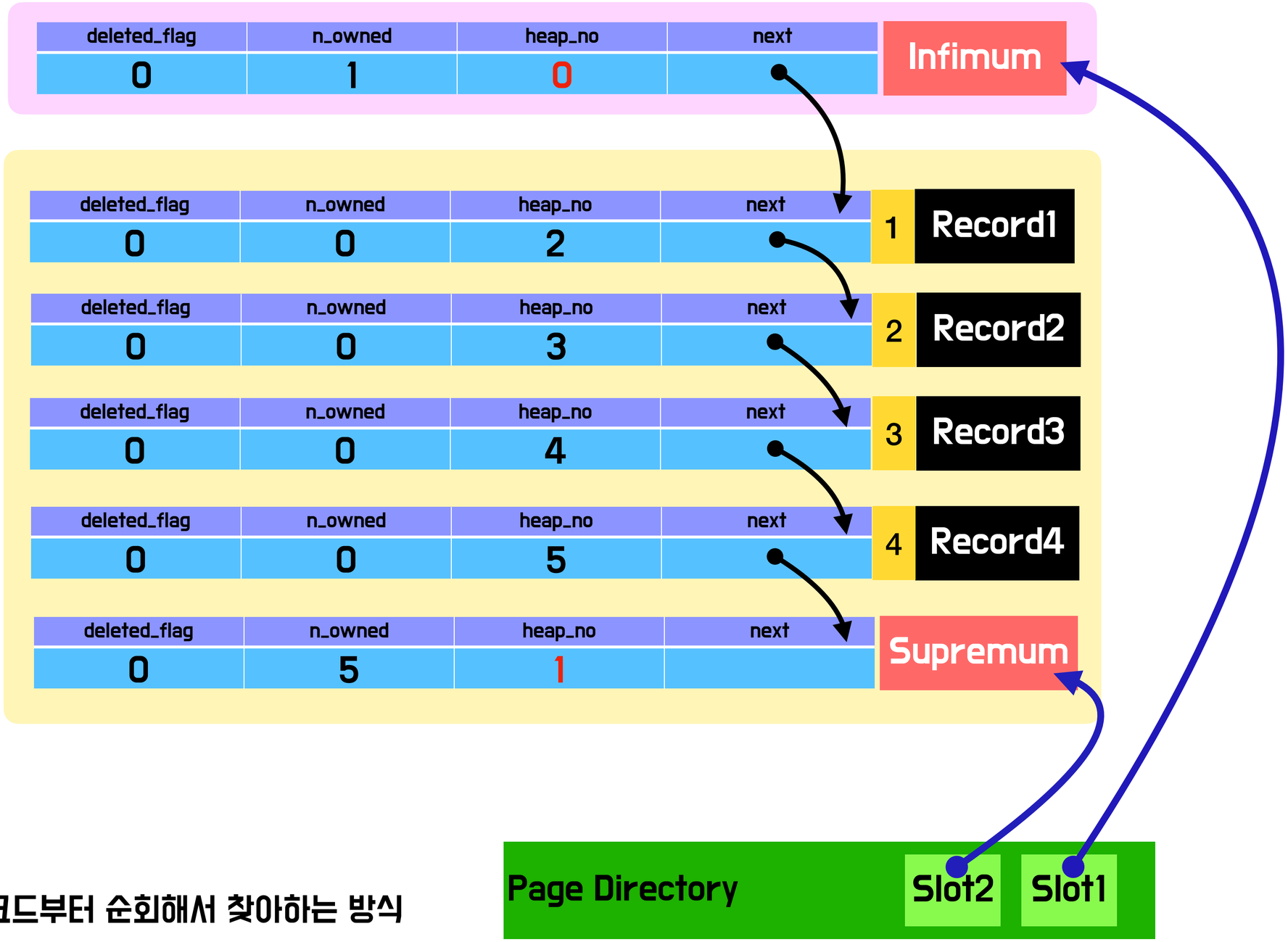
User Records | Free Space



InnoDB Page Structure

Page Directory

1. 기본 구조



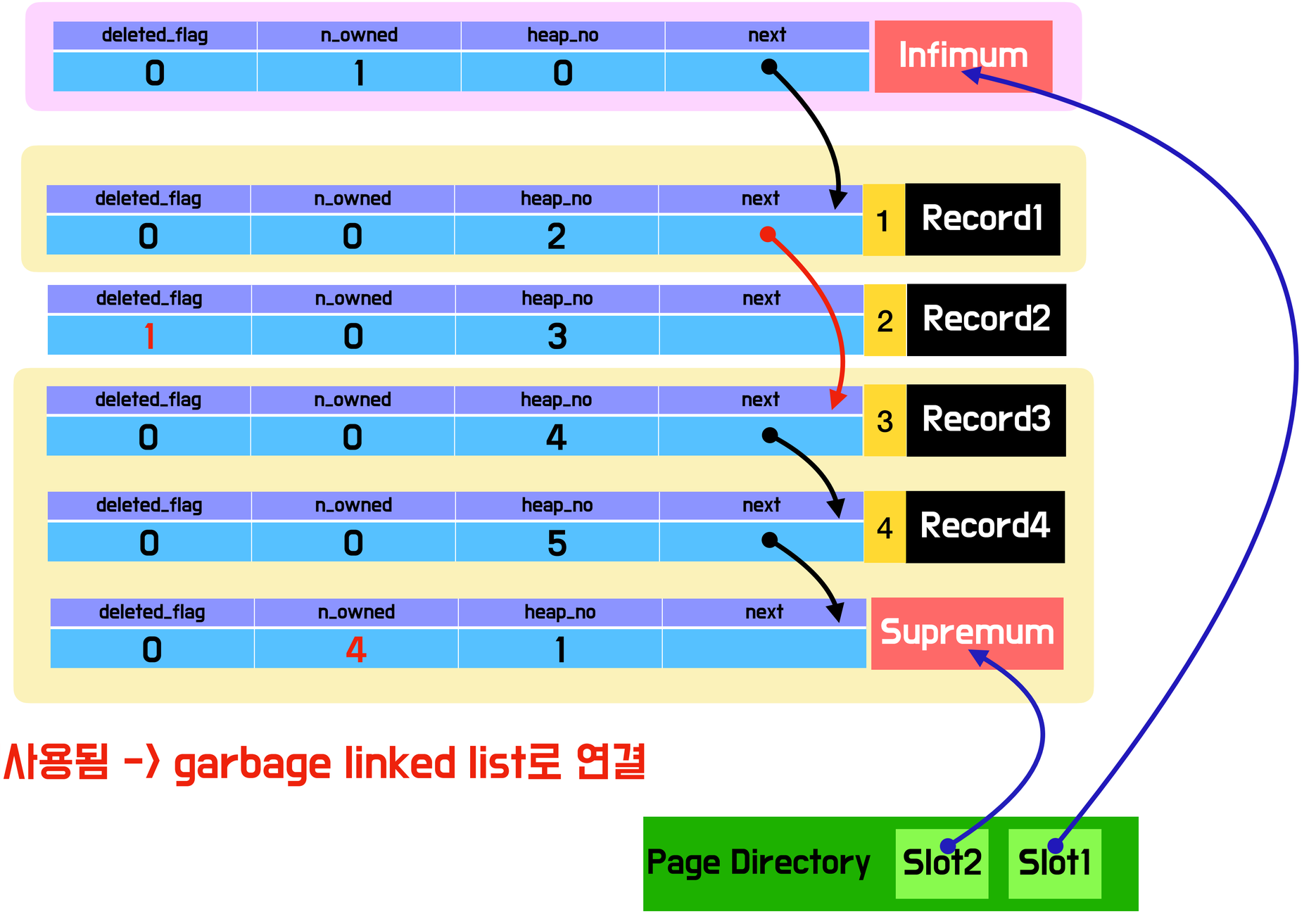
Searching 방식

Slot을 선택해서,
Slot 내부에 PK가 가장 작은 레코드부터 순회해서 찾아하는 방식

InnoDB Page Structure

Page Directory

2. Record delete

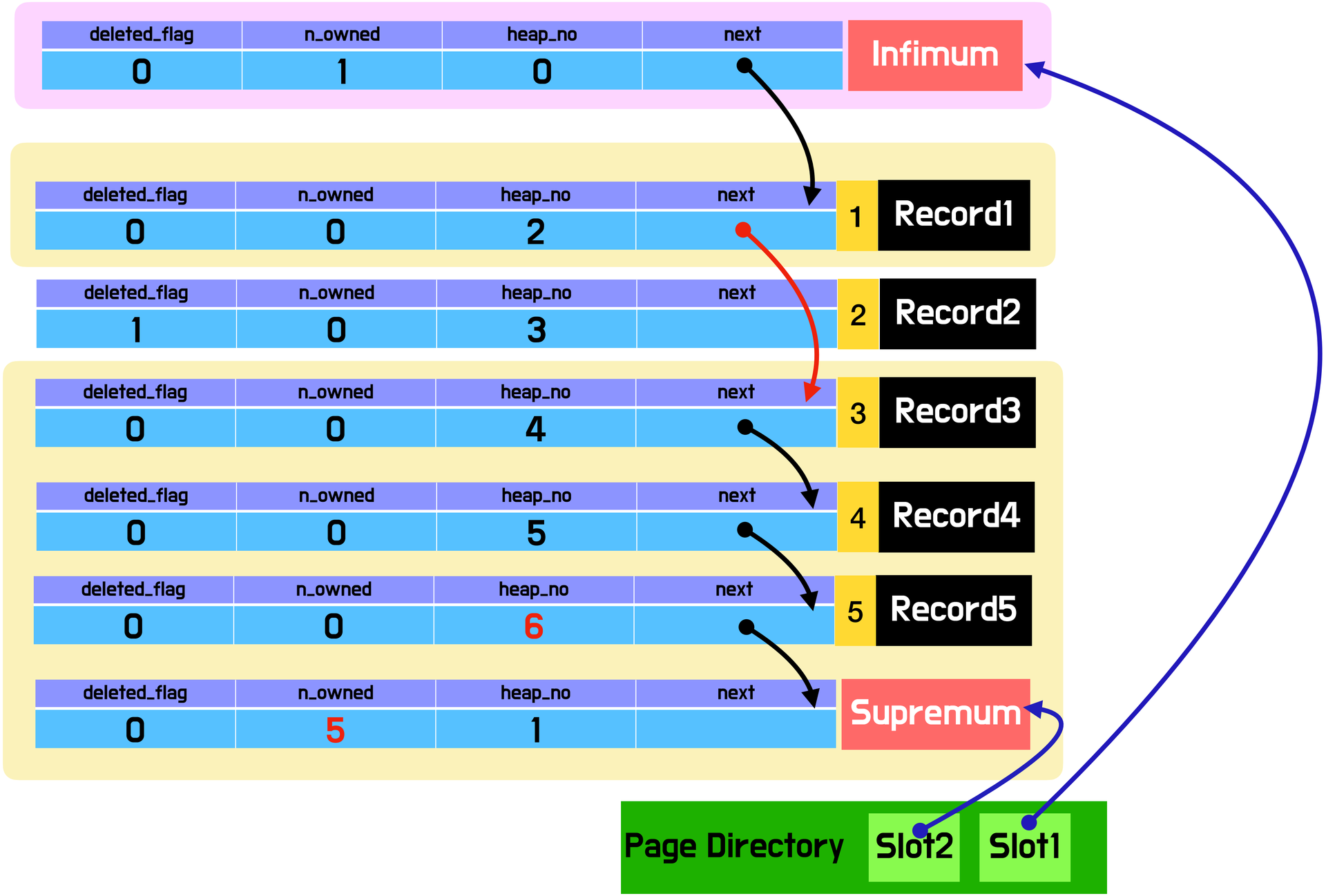


이후에 삭제된 공간은 재사용됨 -> garbage linked list로 연결

InnoDB Page Structure

Page Directory

3. Record insert



InnoDB Page Structure

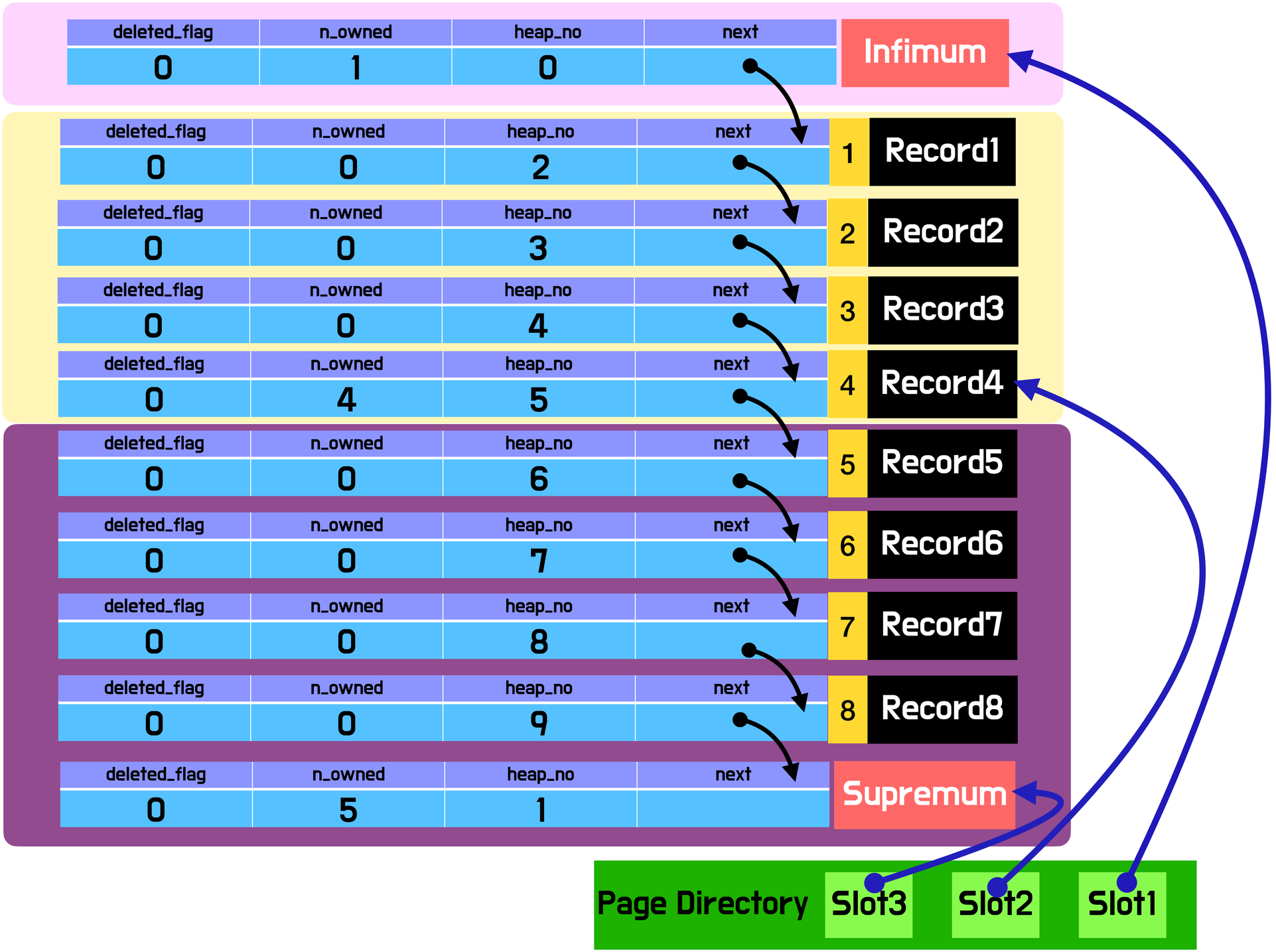
Page Directory

4. Slot 추가

Infimum의 위치를 가지는 slot. 즉 첫번째 슬롯은 infimum만을 가진다 (only 1)

레코드 그룹은 8개의 레코드까지만 가능.

8개 이상의 레코드가 되면 4와 5로 나눠짐 -> 슬롯이 추가됨



InnoDB Page Structure

Fil Trailer

- 데이터 무결성을 위해서 사용되는 부분
- Fil Header의 FIL_PAGE_SPACE, FIL_PAGE_LSN 정보를 가짐
- 첫 4바이트는 FIL_PAGE_SPACE (checksum) 정보를 가짐
- 나머지 4바이트는 FIL_PAGE_LSN 정보를 가짐
- 메모리와 디스크간의 동기화 과정 중 문제가 생겼을 때, 동기화에 실패 했는지 체크하고 복구할 수 있도록 하는 역할