Data Organization in InnoDB

By annamalai.gurusami on Apr 19, 2013

Introduction

This article will explain how the data is organized in InnoDB storage engine. First we will look at the various files that are created by InnoDB, then we look at the logical data organization like tablespaces, pages, segments and extents. We will explore each of them in some detail and discuss about their relationship with each other. At the end of this article, the reader will have a high level view of the data layout within the InnoDB storage engine.

The Files

MySQL will store all data within the data directory. The data directory can be specified using the command line option –data-dir or in the configuration file as datadir. Refer to the Server Command Options for complete details.

By default, when InnoDB is initialized, it creates 3 important files in the data directory – ibdata1, ib_logfile0 and ib_logfile1. The ibdata1 is the data file in which system and user data will be stored. The ib_logfile0 and ib_logfile1 are the redo log files. The location and size of these files are configurable. Refer to Configuring InnoDB for more details.

The data file ibdata1 belongs to the system tablespace with tablespace id (space_id) of 0. The system tablespace can contain more than 1 data file. As of MySQL 5.6, only the system tablespace can contain

more than 1 data file. All other tablespaces can contain only one data file. Also, only the system tablespace can contain more than one table, while all other tablespaces can contain only one table.

The data files and the redo log files are represented in the memory by the C structure fil_node_t.

Tablespaces

By default, InnoDB contains only one tablespace called the system tablespace whose identifier is 0. More tablespaces can be created indirectly using the innodb_file_per_table configuration parameter. In MySQL 5.6, this configuration parameter is ON by default. When it is ON, each table will be created in its own tablespace in a separate data file.

The relationship between the tablespace and data files is explained in the InnoDB source code comment (storage/innobase/fil/fil0fil.cc) which is quoted here for reference:

> "*A tablespace consists of a chain of files. The size of the files does not have to be divisible by the database block size, because we may just leave the last incomplete block unused. When a new file is appended to the tablespace, the maximum size of the file is also specified. At the moment, we think that it is best to extend the file to its maximum size already at the creation of the file, because then we can avoid dynamically extending the file when more space is needed for the tablespace.*"

The last statement about avoiding dynamic extension is applicable only to the redo log files and not the data files. Data files are dynamically extended, but redo log files are pre-allocated. Also, as already mentioned earlier, only the system tablespace can have more than one data file.

It is also clearly mentioned that even though the tablespace can have multiple files, they are thought of as one single large file concatenated together. So the order of files within the tablespace is important.

Pages

A data file is logically divided into equal sized pages. The first page of the first data file is identified with page number of 0, and the next page would be 1 and so on. A page within a tablespace is uniquely identified by the page identifier or page number (page_no). And each tablespace is uniquely identified by the tablespace identifier (space_id). So a page is uniquely identified throughout InnoDB by using the (space_id, page_no) combination. And any location within InnoDB can be uniquely identified by the (space_id, page_no, page_offset) combination, where page_offset is the number of bytes within the given page.

How the pages from different data files relate to one another is explained in another source code comment: *"A block's position in the tablespace is specified with a 32-bit unsigned integer. The files in the chain are thought to be catenated, and the block corresponding to an address n is the nth block in the catenated file (where the first block is named the 0th block, and the incomplete block fragments at the end of files are not taken into account). A tablespace can be extended by appending a new file at the end of the chain."* This means that the first page of all the data files will not have page_no of 0 (zero). Only the first page of the first data file in a tablespace will have the page_no as 0 (zero).

Also in the above comment it is mentioned that the page_no is a 32-bit unsigned integer. This is the size of the page_no when stored on the disk.

Every page has a page header (page_header_t). For more details on this please refer to the Jeremy Cole's blog "The basics of InnoDB space file layout."

Extents

An extent is 1MB of consecutive pages. The size of one extent is defined as follows (1048576 bytes = 1MB):

```
#define FSP_EXTENT_SIZE (1048576U / UNIV_PAGE_SIZE)
```

The macro UNIV_PAGE_SIZE used to be a compile time constant. From mysql-5.6 onwards it is a global variable. The number of pages in an extent depends on the page size used. If the page size is 16K (the default), then an extent would contain 64 pages.

Page Types

One page can be used for many purposes. The page type will identify the purpose for which the page is being used. The page type of each page will be stored in the page header. The page types are available in the header file storage/innobase/include/fil0fil.h. The following table provides a brief description of the page types.

Page Type	Description
FIL_PAGE_INDEX	The page is a B-tree node
FIL_PAGE_UNDO_LOG	The page stores undo logs
FIL_PAGE_INODE	contains an array of fseg_inode_t objects.
FIL_PAGE_IBUF_FREE_LIST	The page is in the free list of insert buffer or change buffer.
FIL_PAGE_TYPE_ALLOCATED	Freshly allocated page.
FIL_PAGE_IBUF_BITMAP	Insert buffer or change buffer bitmap
FIL_PAGE_TYPE_SYS	System page
FIL_PAGE_TYPE_TRX_SYS	Transaction system data
FIL_PAGE_TYPE_FSP_HDR	File space header
FIL_PAGE_TYPE_XDES	Extent Descriptor Page
FIL_PAGE_TYPE_BLOB	Uncompressed BLOB page
FIL_PAGE_TYPE_ZBLOB	First compressed BLOB page

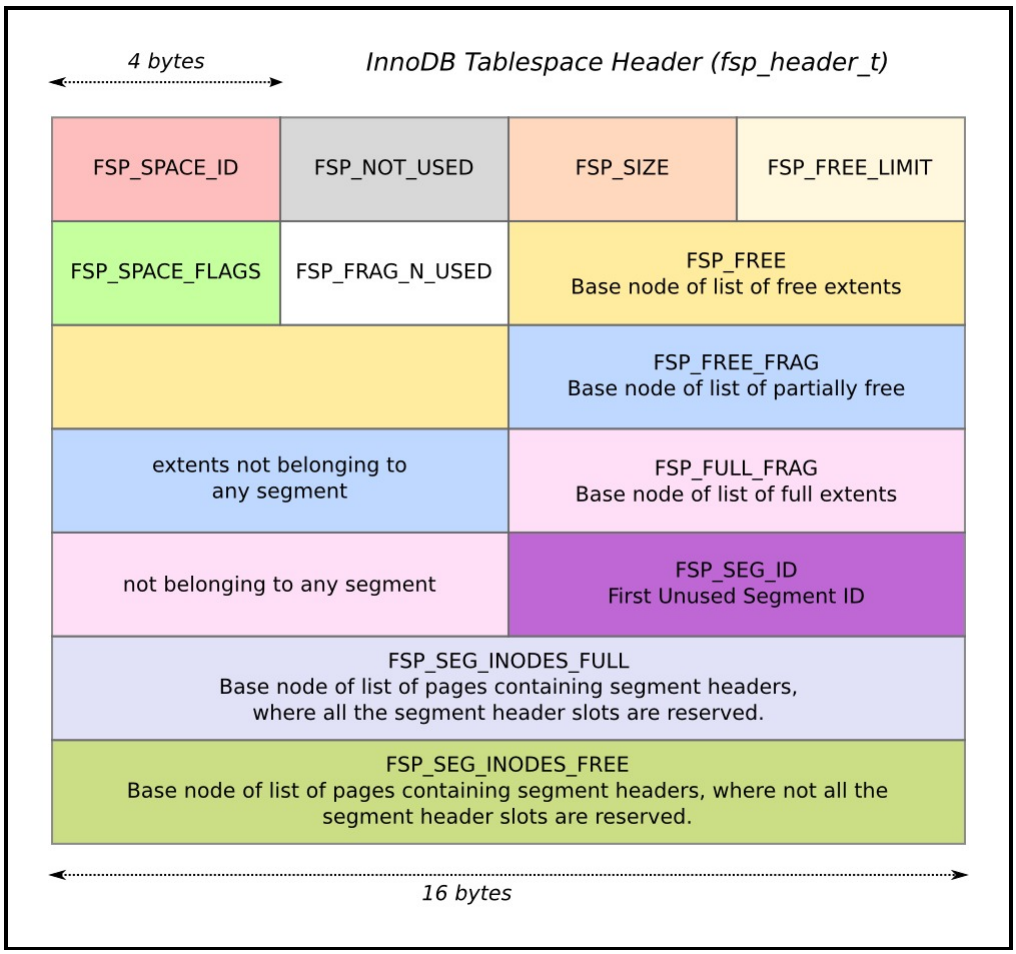
Page Type	Description
FIL_PAGE_TYPE_ZBLOB2	Subsequent compressed BLOB page

Each page type is used for different purposes. It is beyond the scope of this article, to explore each page type. For now, it is sufficient to know that all pages have a page header (page_header_t) and they store the page type in it, and based on the page type the contents and the layout of the page would be decided.

Tablespace Header

Each tablespace will have a header of type fsp_header_t. This data structure is stored in the first page of a tablespace.

- The table space identifier (space_id)
- Current size of the table space in pages.
- List of free extents
- List of full extents not belonging to any segment.
- List of partially full/free extents not belonging to any segment.
- List of pages containing segment headers, where all the segment inode slots are reserved. (pages of type FIL_PAGE_INODE)
- List of pages containing segment headers, where not all the segment inode slots are reserved. (pages of type FIL_PAGE_INODE).



From the tablespace header, we can access the list of segments available in the tablespace. The total space occupied by the tablespace header is given by the macro FSP_HEADER_SIZE, which is equal to 16*7 = 112 bytes.

Reserved Pages of Tablespace

As mentioned earlier, InnoDB will always contain one tablespace called the system tablespace with identifier 0. This is a special tablespace and is always kept open as long as the MySQL server is running. The first few pages of the system tablespace is reserved for internal usage. This information can be obtained from the header storage/innobase/include/fsp0types.h. They are listed below with a short description.

Page Number	***The Page Name***	***Description***
0	FSP_XDES_OFFSET	The extent descriptor page.
1	FSP_IBUF_BITMAP_OFFSET	The insert buffer bitmap page.
2	FSP_FIRST_INODE_PAGE_NO	The first inode page number.

Page Number	*The Page Name*	*Description*
3	FSP_IBUF_HEADER_PAGE_NO	Insert buffer header page in system tablespace.
4	FSP_IBUF_TREE_ROOT_PAGE_NO	Insert buffer B-tree root page in system tablespace.
5	FSP_TRX_SYS_PAGE_NO	Transaction system header in system tablespace.
6	FSP_FIRST_RSEG_PAGE_NO	First rollback segment page, in system tablespace.
7	FSP_DICT_HDR_PAGE_NO	Data dictionary header page in system tablespace.

As can be noted from above, the first 3 pages will be there in any tablespace. But the last 5 pages are reserved only in the case of system tablespace. In the case of other tablespaces only 3 pages are reserved.

When the option innodb_file_per_table is enabled, then for each table a separate tablespace with one data file would be created. The source code comment in the function dict_build_table_def_step() states the following:

```
/* We create a new single-table tablespace for the table.
We initially let it be 4 pages:
- page 0 is the fsp header and an extent descriptor page,
- page 1 is an ibuf bitmap page,
- page 2 is the first inode page,
- page 3 will contain the root of the clustered index of the
table we create here. */
```

File Segments

A tablespace can contain many file segments. File segments (or just segments) is a logical entity. Each segment has a segment header (fseg_header_t), which points to the inode (fseg_inode_t) describing the file segment. The file segment header contains the following information:

- The space to which the inode belongs
- The page_no of the inode
- The byte offset of the inode
- The length of the file segment header (in bytes).

Note: It would have been really more readable (at source code level) if fseg_header_t and fseg_inode_t had proper C-style structures defined for them.

The fseg_inode_t object contains the following information:

- The segment id to which it belongs.
- List of full extents.
- List of free extents of this segment.
- List of partially full/free extents
- Array of individual pages belonging to this segment. The size of this array is half an extent.

When a segment wants to grow, it will get free extents or pages from the tablespace to which it belongs.

Table

In InnoDB, when a table is created, a clustered index (B-tree) is created internally. This B-tree contains two file segments, one for the non-leaf pages and the other for the leaf pages. From the source code documentation:

> *"In the root node of a B-tree there are two file segment headers. The leaf pages of a tree are allocated from one file segment, to make them consecutive on disk if possible. From the other file segment we allocate pages for the non-leaf levels of the tree."*

For a given table, the root page of a B-tree will be obtained from the data dictionary. So in InnoDB, each table exists within a tablespace, and contains one B-tree (the clustered index), which contains 2 file segments. Each file segment can contain many extents, and each extent contains 1MB of consecutive pages.

Conclusion

This article discussed the details about the data organization within InnoDB. We first looked at the files created by InnoDB, and then discussed about the various logical entities like tablespaces, pages, page types, extents, segments and tables. We also looked at the relationship between each one of them.

Send in your comments and feedback to annamalai.gurusami@oracle.com.

Category: InnoDB

Tags: data extents segments tablespace

Permanent link to this entry

« Repeatable Read... | Main | Introduction to... »
Comments:

It was a very good blog about the internals. If possible please share the internals InnoDB Log file (Redo Log).

Thanks,
Karthik.P.R

Posted by **karthik P R** on May 09, 2013 at 09:49 AM EDT #

Hi Karthik,

Thanks for your feedback. I've made a note of your request. Either me or someone from our team will explain about the redo log format.

Rgds,
anna

Posted by **Annamalai Gurusami (அண்ணாமலை குருசாமி)** on May 09, 2013 at 11:34 PM EDT #

Post a Comment:
Comments are closed for this entry.

About

This is the InnoDB team blog.

Search

Enter search term:
Submit Search
Search only this blog

Recent Posts

- Transaction life cycle improvements in 5.7.3
- MySQL 5.7.3: Deep dive into 1mil QPS with InnoDB & Memcached
- InnoDB Temporary Tables just got faster
- InnoDB 5.7 performance improvements
- InnoDB Redundant Row Format
- Redo Logging in InnoDB
- Introduction to Transaction Locks in InnoDB Storage Engine
- Data Organization in InnoDB
- Repeatable Read Isolation Level in InnoDB - How Consistent Read View Works
- Online ALTER TABLE in MySQL 5.6

Top Tags

- alter
- column
- compression
- consistent
- create
- data
- ddl
- drop
- extents
- failure
- format
- fts
- index
- innodb
- isolation
- level
- locks
- logging
- memcached
- mvcc
- online
- performance
- plugin
- read
- redo
- repeatable
- row
- segments
- snapshot
- stats
- table
- tables
- tablespace
- temporary
- transaction
- view
- wal

Categories

- Features
- InnoDB
- MySQL
- News
- Performance

- Private
- Tips

Archives

« December 2016

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Today

Bookmarks

- Events
- MySQL Support Blogs
- Oracle Technology Network
- Oracle Technology Network Blog
- Planet MySQL
- Recent Articles
- Technology Newsletters
- The State of the Dolphin

Menu

- Blogs Home
- Weblog
- Login

Feeds

RSS

- All
- /Features
- /InnoDB
- /MySQL
- /News
- /Performance
- /Private
- /Tips
- Comments

Atom

- All
- /Features
- /InnoDB
- /MySQL
- /News
- /Performance
- /Private
- /Tips
- Comments