Data Organization in InnoDB

— by royalwzy

Personal Profile

Database technology enthusiast

Oracle 10g/11g OCM
MySQL 5.6 Database Administrator
MCITP/MCTS DBA(MSSQL2008)
RHCE
Java Programmer

Database technology researcher

mysqlexp
mysqlclone

Database technology user

More than 5 years experience in database use and management
Proficient in synchronizing data between heterogeneous database
Skilled in designing and optimizing database
Install and configure MySQL cluster and high-availability cluster environment
Synchronize and merge data from Oracle to MySQL

Database technology sharer

Oracle ACEA, Oracle Young Expert
ACOUG/SHOUG/OCMU core member
ITPUB Moderator and Community Expert
Senior lecturer in "ENMO" and "UPLOOKING"
Active in OTN & ITPUB and other forums
My blog:royalwzy.com

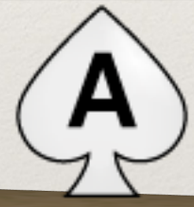


Table of Contents

- Various files that are created by InnoDB
- Logical data organization like tablespaces, pages, segments and extents
- Explore each of them in some detail and discuss about their relationship with each other
- Data layout within the InnoDB storage engine

The Files in InnoDB

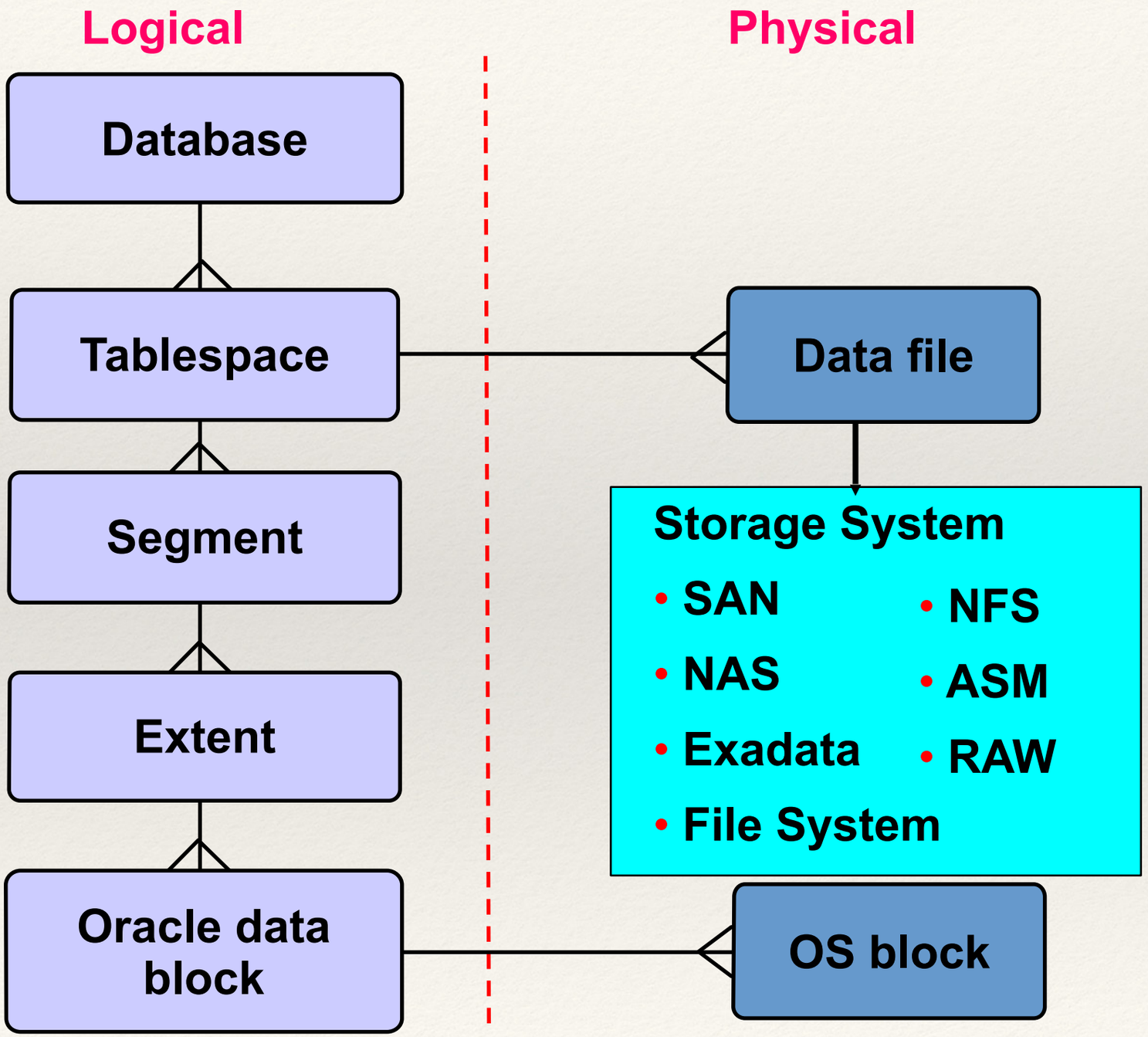
- MySQL will store all data within the data directory
- By default, when InnoDB is initialized, it creates 3 important files
- As of MySQL 5.6, only the system tablespace can contain more than 1 data file. All other tablespaces can contain only one data file
- The data files and the redo log files are represented in the memory by the C structure fil_node_t

Logical data organization

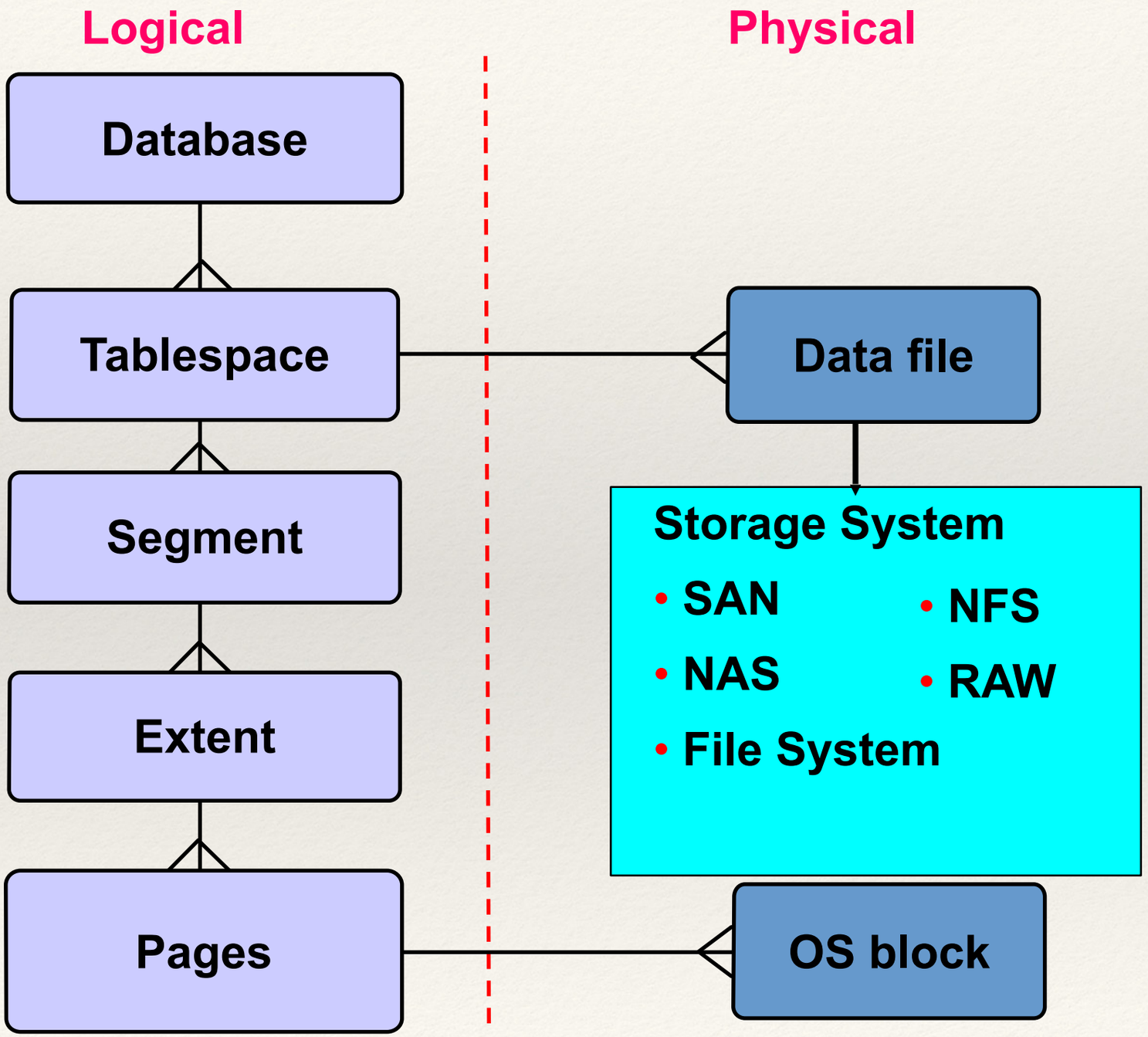
- Tablespaces
- Pages
- Extents
- File Segments

> *“Our tablespace concept is similar to the one of Oracle”*

Logical and Physical Database Structures in Oracle



Logical and Physical Database Structures in InnoDB



Tablesapce

- innodb_file_per_table configuration parameter
- What's the relationship between the tablespace and data files

> *“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.”*

Tablespace Header

- Each tablespace will have a header of type fsp_header_t
  - 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
  - List of pages containing segment headers, where not all the segment inode slots are reserved
- From the tablespace header, we can access the list of segments available in the tablespace

4 bytes

InnoDB Tablespace Header (fsp_header_t)

FSP_SPACE_ID	FSP_NOT_USED	FSP_SIZE	FSP_FREE_LIMIT
FSP_SPACE_FLAGS	FSP_FRAG_N_USED	FSP_FREE Base node of list of free extents	
		FSP_FREE_FRAG Base node of list of partially free	
extents not belonging to any segment		FSP_FULL_FRAG Base node of list of full extents	
not belonging to any segment		FSP_SEG_ID First Unused Segment ID	
FSP_SEG_INODES_FULL Base node of list of pages containing segment headers, where all the segment header slots are reserved.			
FSP_SEG_INODES_FREE Base node of list of pages containing segment headers, where not all the segment header slots are reserved.			

16 bytes

Pages

- A data file is logically divided into equal sized pages
- How the pages from different data files relate to one another
- Every page has a page header

> *"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."*

Page Types

Page Type	Value	Description
FIL_PAGE_INDEX	17855	The page is a B-tree node
FIL_PAGE_UNDO_LOG	2	The page stores undo logs
FIL_PAGE_INODE	3	contains an array of fseg_inode_t objects.
FIL_PAGE_IBUF_FREE_LIST	4	The page is in the free list of insert buffer or change buffer
FIL_PAGE_TYPE_ALLOCATED	0	Freshly allocated page.
FIL_PAGE_IBUF_BITMAP	5	Insert buffer or change buffer bitmap
FIL_PAGE_TYPE_SYS	6	System page
FIL_PAGE_TYPE_TRX_SYS	7	Transaction system data
FIL_PAGE_TYPE_FSP_HDR	8	File space header
FIL_PAGE_TYPE_XDES	9	Extent Descriptor Page
FIL_PAGE_TYPE_BLOB	10	Uncompressed BLOB page
FIL_PAGE_TYPE_ZBLOB	11	First compressed BLOB page
FIL_PAGE_TYPE_ZBLOB2	12	Subsequent compressed BLOB page
FIL_PAGE_TYPE_LAST	FIL_PAGE_TYPE_ZBLOB2	Last page type

Extents

- An extent is 1MB of consecutive pages
- #define FSP_EXTENT_SIZE (1048576U / UNIV_PAGE_SIZE)

File Segments

- A tablespace can contain many file segments
- Information of segment header
  - 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)

Table

- In InnoDB, when a table is created, a clustered index (B-tree) is created internally
- The root page of a B-tree will be obtained from the data dictionary

> *"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."*

Q & A