InnoDB Internals

Calvin Sun

Senior Manager, Twitter

WHO AM I?

- Calvin Sun (孙春生), email: csun@twitter.com
- Joined Twitter in Mar 2013
- Senior Manager at Oracle, Feb 2008 - Mar 2013
- Manager at MySQL, Jan 2006 - Jan 2008
- Team lead, product architect at Pervasive Software, Jul 1997 - Dec 2005
- MS in computer science from USTC (中国科

Agenda

- Introduction to InnoDB
- InnoDB Data Format
- InnoDB Logging
- InnoDB Execution
- InnoDB Online Operations
- InnoDB Monitoring & Diagnostics
- Get Involved!

Introduction to InnoDB


facebook


Google


YAHOO!


Alibaba.com
Global trade starts here.™


淘宝网
Taobao.com


flickr
from YAHOO!


craigslist

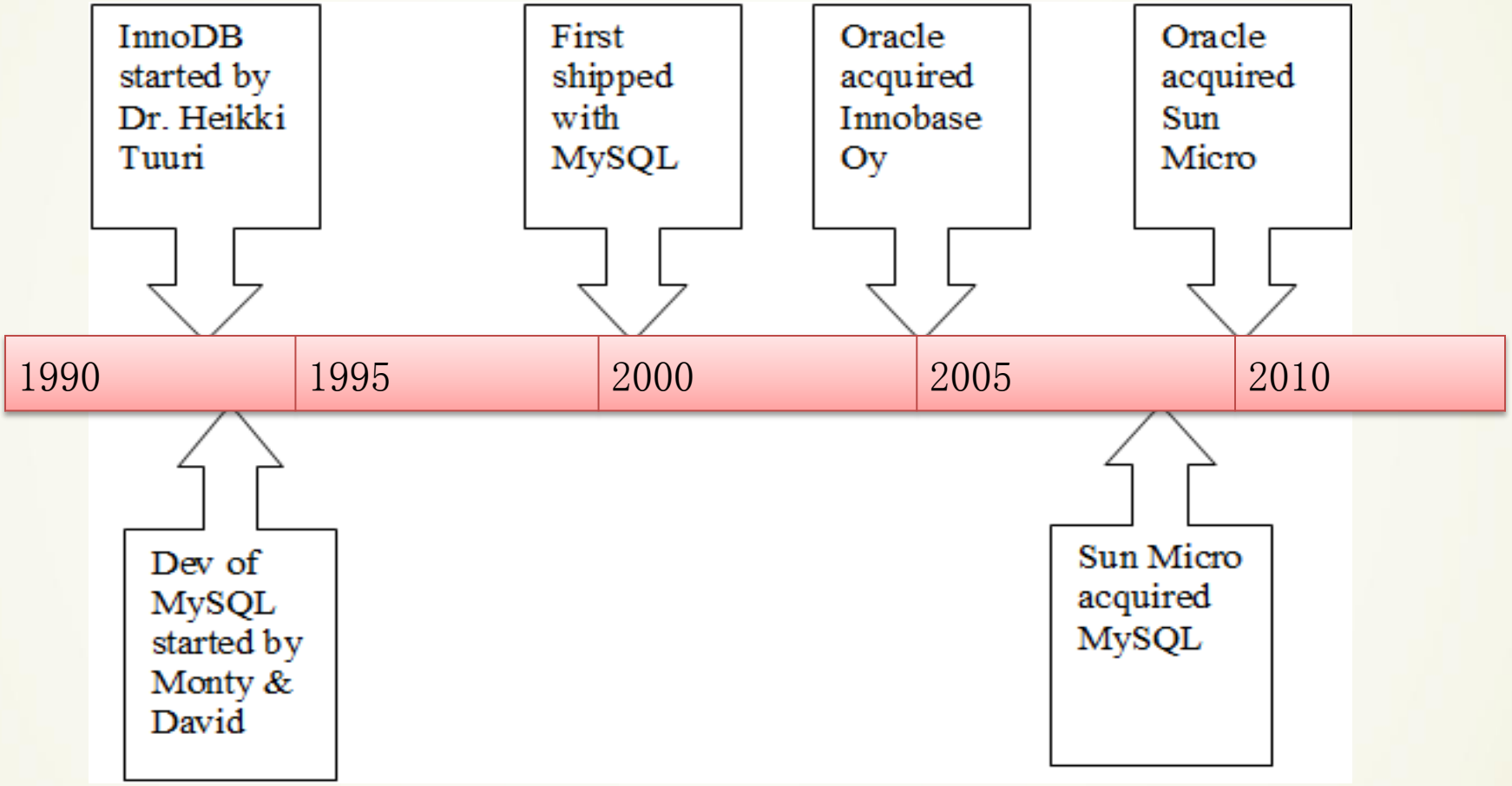

twitter


WIKIPEDIA
The Free Encyclopedia

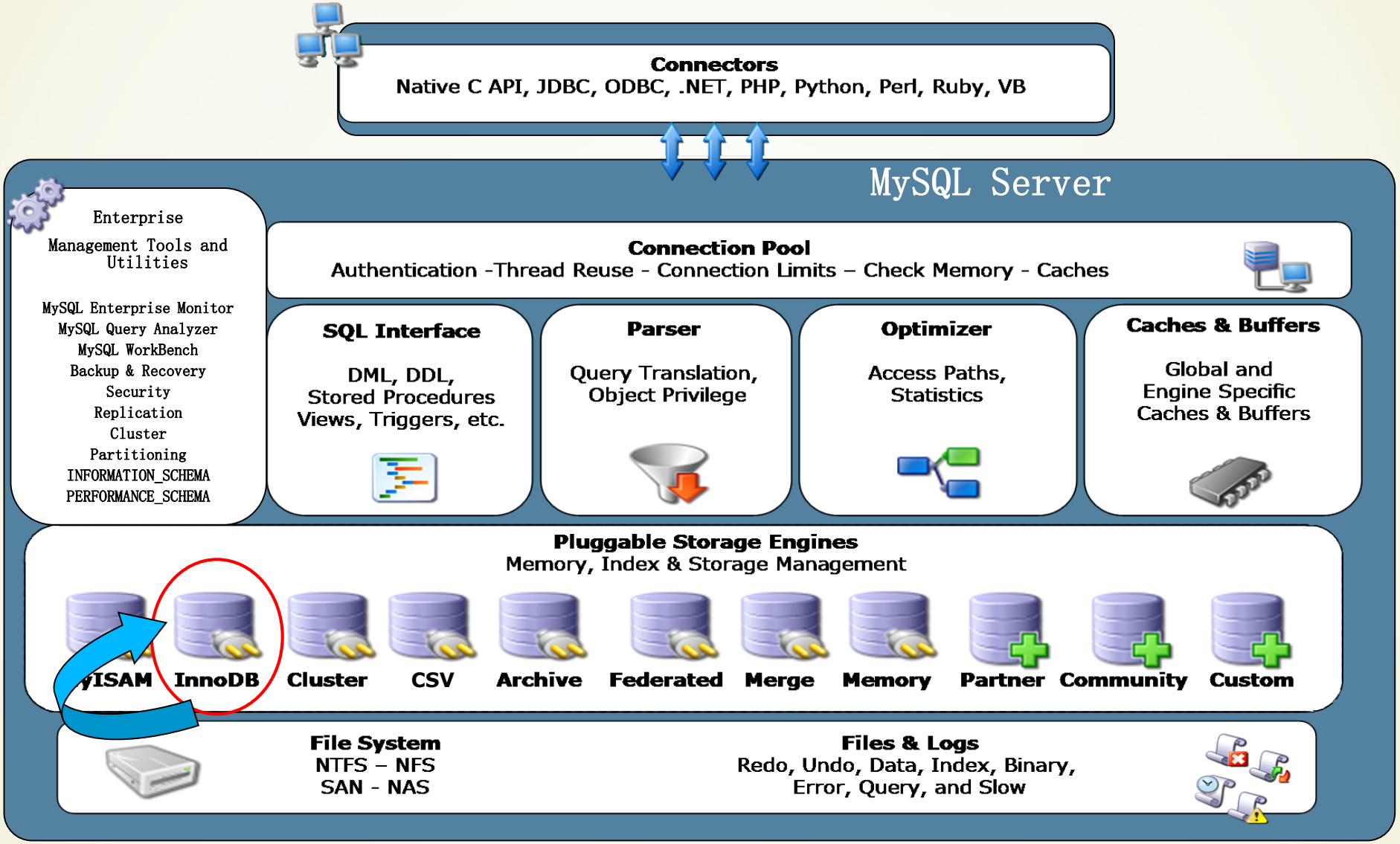

RIGHT
NOW
TECHNOLOGIES


YouTube

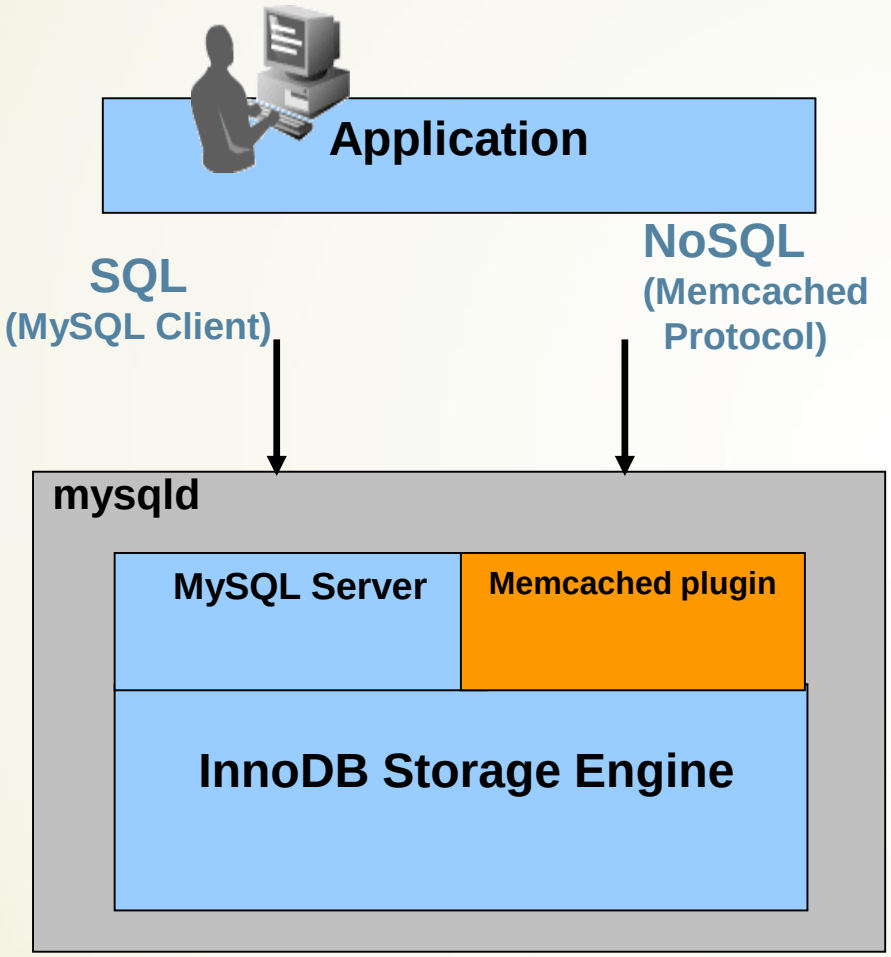
InnoDB Timeline



MySQL Server Architecture



NoSQL to InnoDB via Memcached API



- Fast, simple access to InnoDB
  - Accessed via Memcached API
  - Use existing Memcached clients
  - Bypasses SQL transformations
- SQL/NoSQL access
  - NoSQL for key-value operations
  - SQL for rich queries, JOINs, FKs, etc.
- Implementation
  - Memcached daemon plug-in to mysqld
  - Memcached protocol mapped to the native InnoDB API
  - Shared process space for ultra-low latency

InnoDB Features - Transactions

- Full transaction support
  - Atomicity
  - Consistency
  - Isolation
  - Durability
- SQL-standard isolation levels
- Row-level locking
- Multi-version concurrency control (MVCC)
- Automatic deadlock detection
- Plus
  - Automatic crash recovery
  - Referential integrity

InnoDB Design Considerations

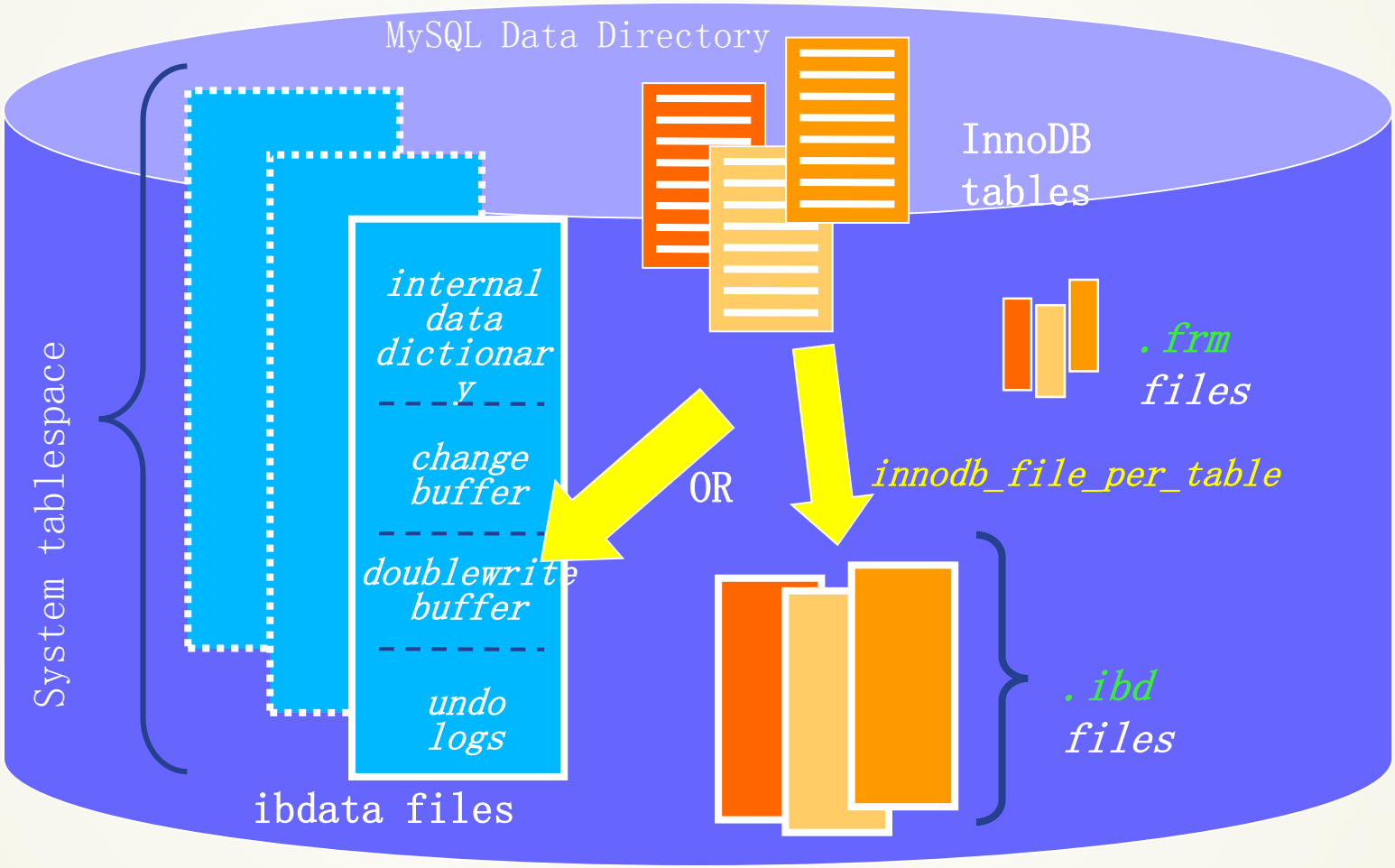
- Modeled on Gray & Reuter's "*Transaction Processing: Concepts & Techniques*"
  - Next key locking
- Also emulated the Oracle architecture
  - Multi-version concurrency control (MVCC)
  - Undo info in the database, not the logs
  - Tablespaces for data & index storage
- Added unique subsystems/features

InnoDB Innovative Features

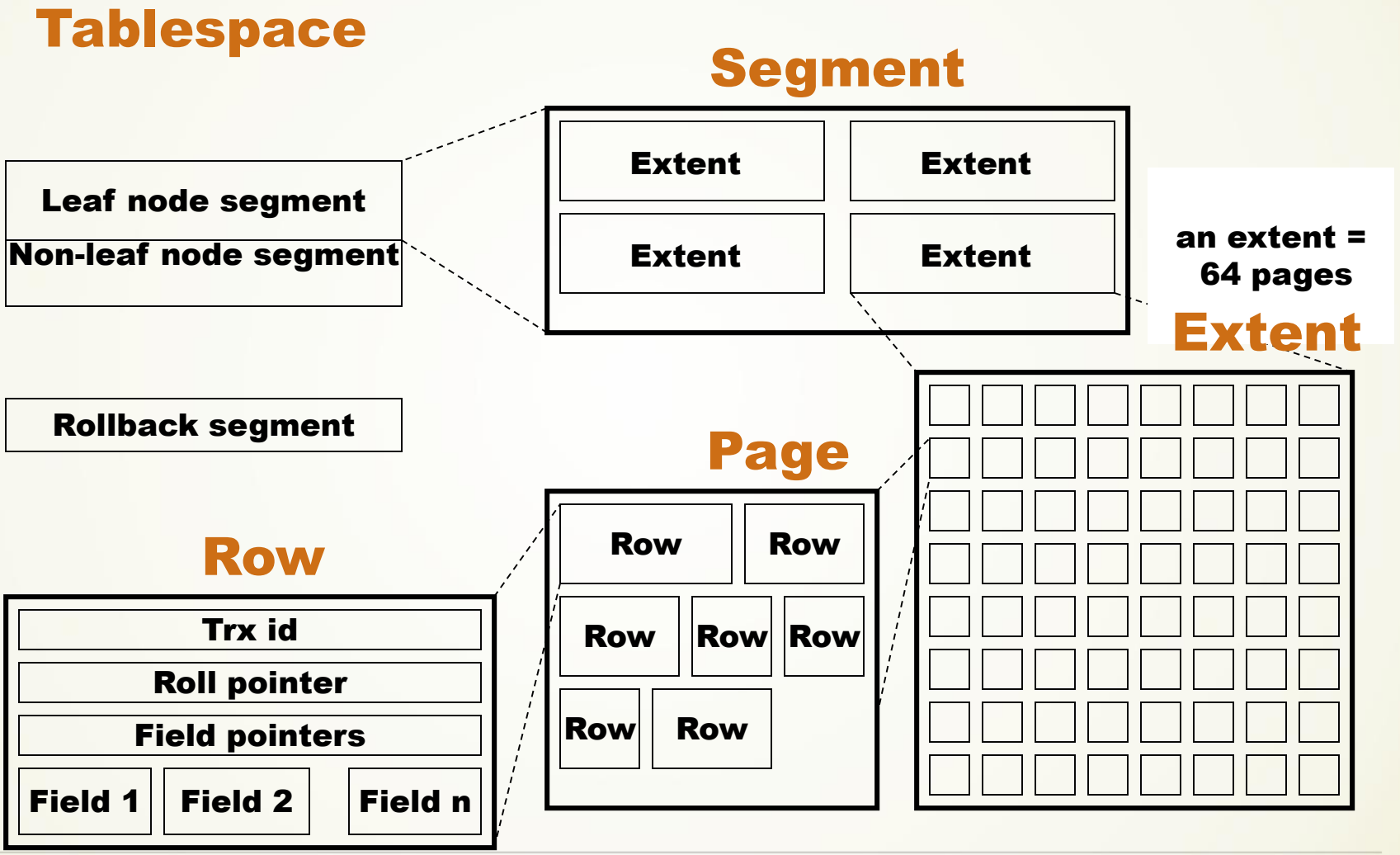
- **Adaptive Hash Indexes**: automatically created on prefix of key for frequent queries
  - Approximates in-memory databases
- **Change Buffering**: buffers modifications to secondary indexes when the leaf pages are not in the buffer pool
  - Batched merges result in less random access patterns
- **Doublewrite Buffer**: data first written into the buffer, then flush to the datafiles
  - Preventing partially written pages

InnoDB Data Format

InnoDB Database Files

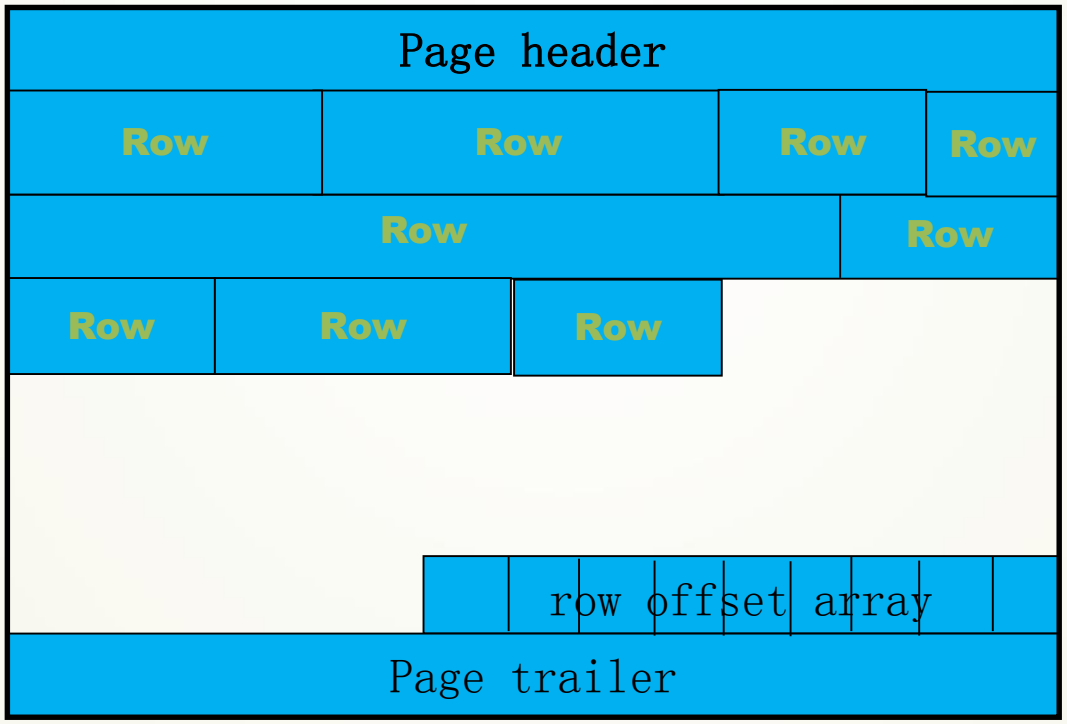


InnoDB Tablespaces

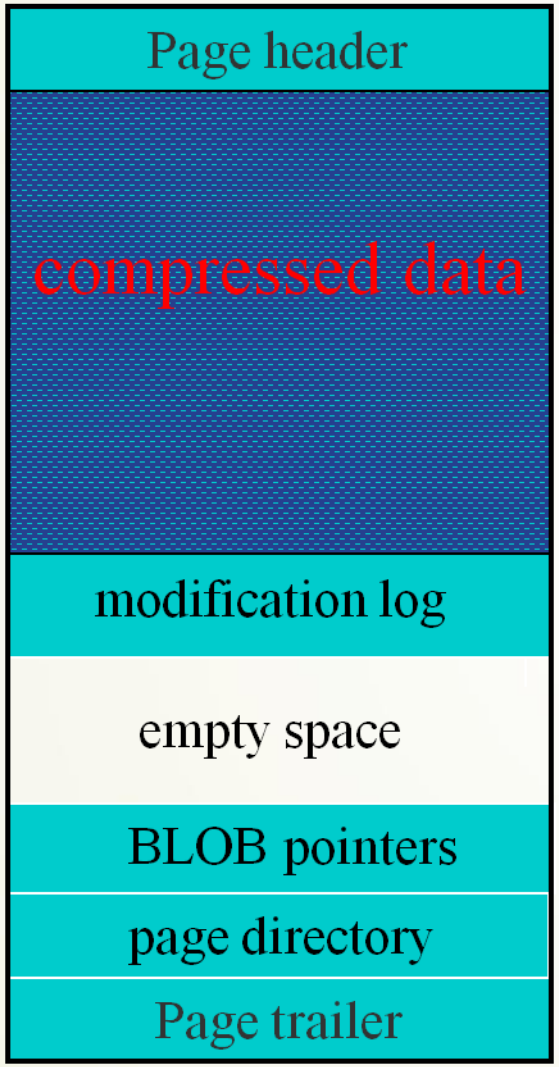


InnoDB Page Structure

A page consists of: a page header, a page trailer, and a page body (rows or other contents).



InnoDB Compressed Pages

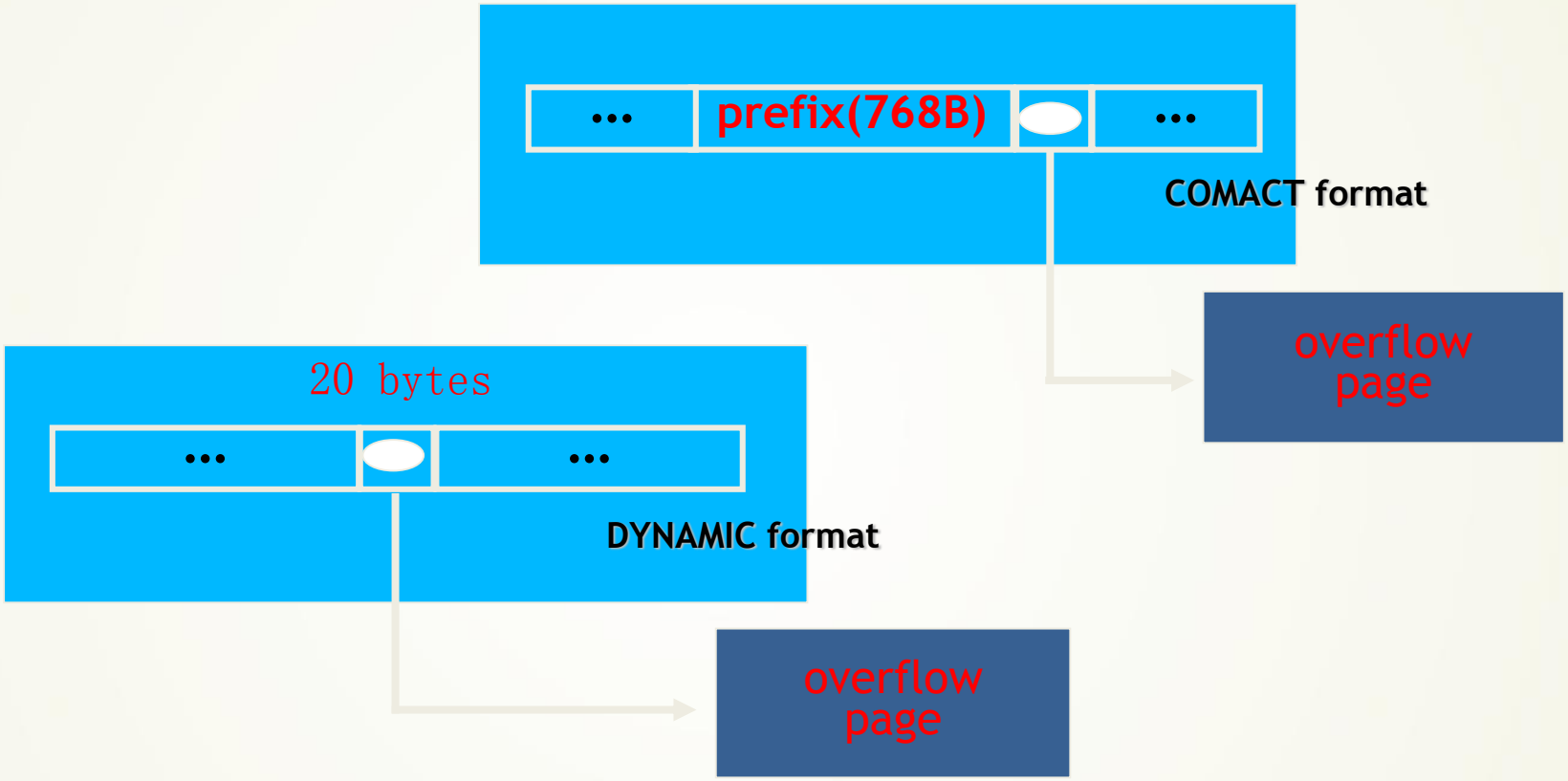


- InnoDB keeps a "modification log" in each page
- Updates & inserts of small records are written to the log w/o page reconstruction; deletes don't even require uncompression
- Log also tells InnoDB if the page will compress to fit page size
- When log space runs out, InnoDB uncompresses the page, applies the changes and recompresses the page

InnoDB Row Format

- InnoDB Row Format:
  - **Redundant**: The oldest InnoDB row format
  - **Compact**: The default InnoDB row format since MySQL 5.0.3. It has a more compact representation for nulls and variable-length fields
  - **Dynamic**: Store long columns entirely “off-page”.
  - **Compress**: Compress data & index pages from normal page size (16KB) to specified compressed page size

InnoDB Row Structure



Record hdr	Trx ID	Roll ptr	Fld ptrs	overflow-page ptr	.. Field values

InnoDB Logging

ARIES

ARIES, *Algorithms for Recovery and Isolation Exploiting Semantics*, is a recovery algorithm used by almost all modern database systems.

Three main principles lie behind ARIES:

- Write ahead logging (WAL)
- Repeating history during Redo
- Logging changes during Undo

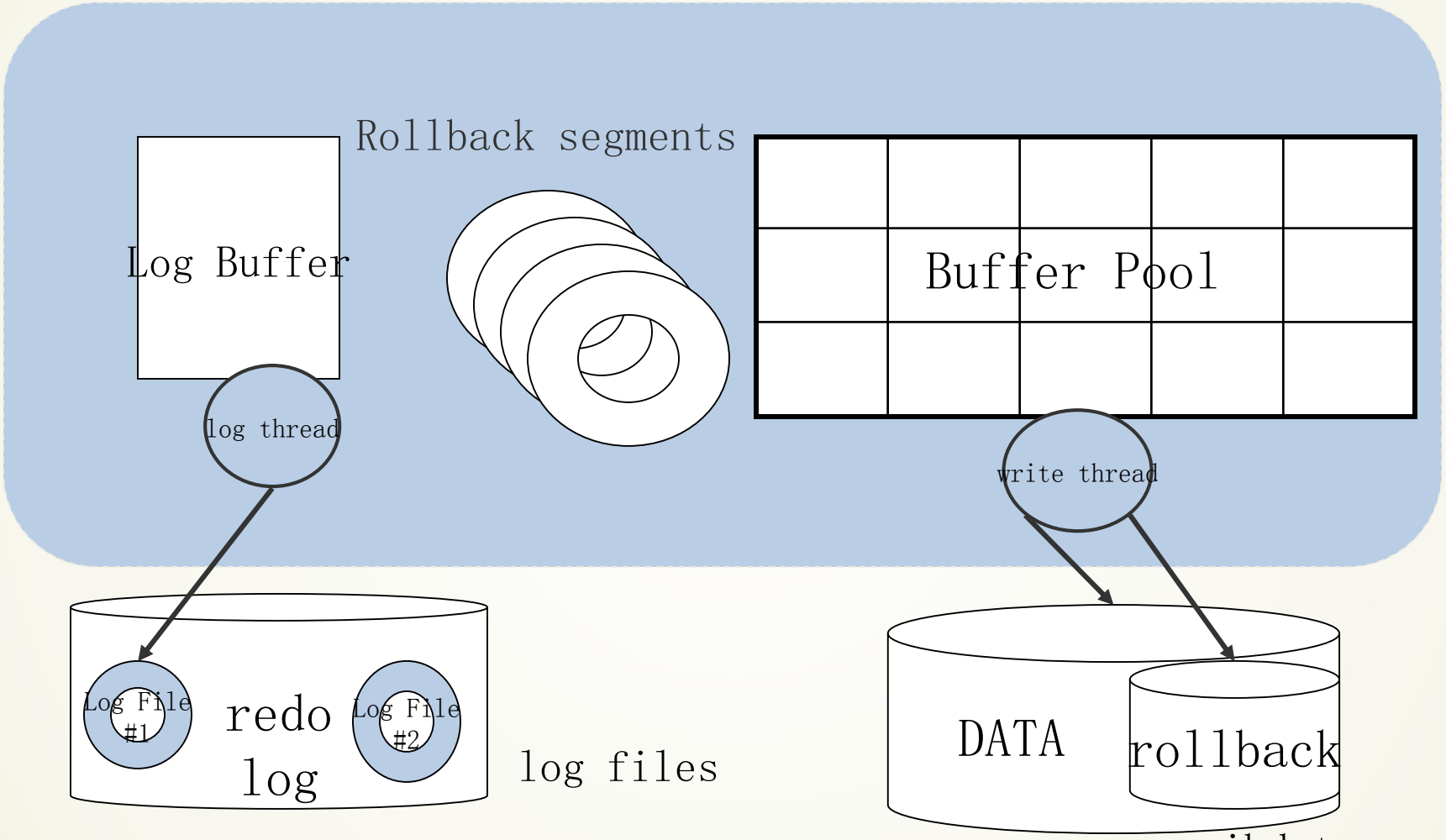
Types of Logging

- **Physical logging**: changes to data pages and their data record are logged by their byte offsets and byte-for-byte copies of the data.
- **Logical logging**: page and byte locations do not matter; only the logical operations are recorded in the recovery log.
- **Physiological logging**: physical to a page, logical within a page, and it codes the page operation in a concise way.

Why Physiological Logging?

- Smaller log files
- Quick recovery
- Proven: It is the method of choice in modern database systems
- Address fundamental flaw in logical logging: **operations are not atomic.**
  - Example – “insert t in T ” requires an update to both a data and an index page. A crash might occur after t has been inserted in T but before the index has been updated
  - Example – page split

InnoDB Logging



InnoDB Redo Log

- Physiological logging
- The redo log remembers EVERY operation on any page in the database
- Redo log record format:

SpaceID	PageNo	Offset	OperationType	Changes on that page

- Changes (only redo values, no old values) except for DELETEs, which need no change notes at all
- Examples of operations
  - Insert after record at offset 5444
  - Reorganize page 1234

InnoDB Undo Log

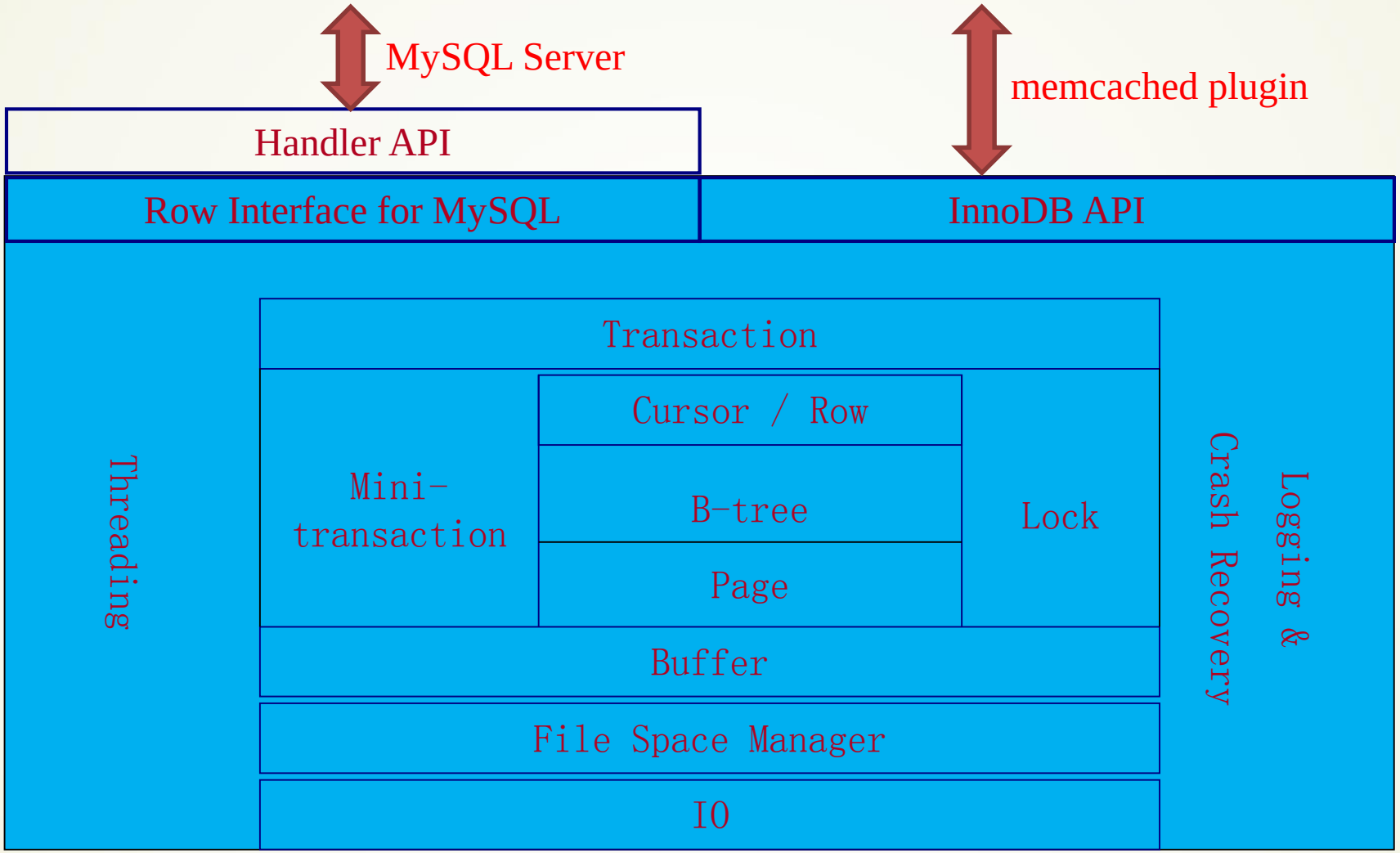
A collection of undo log records

Primary Key Value	Old trx id	Old values on that row

- **Primary Key Value** (no page numbers, no physical addresses)
- **Old transaction ID:** The ID of the trx that updated that row
- The **old field values** of that row, which will make the old transaction ID the newest update to the row in question

InnoDB Execution

InnoDB Architecture: Execution



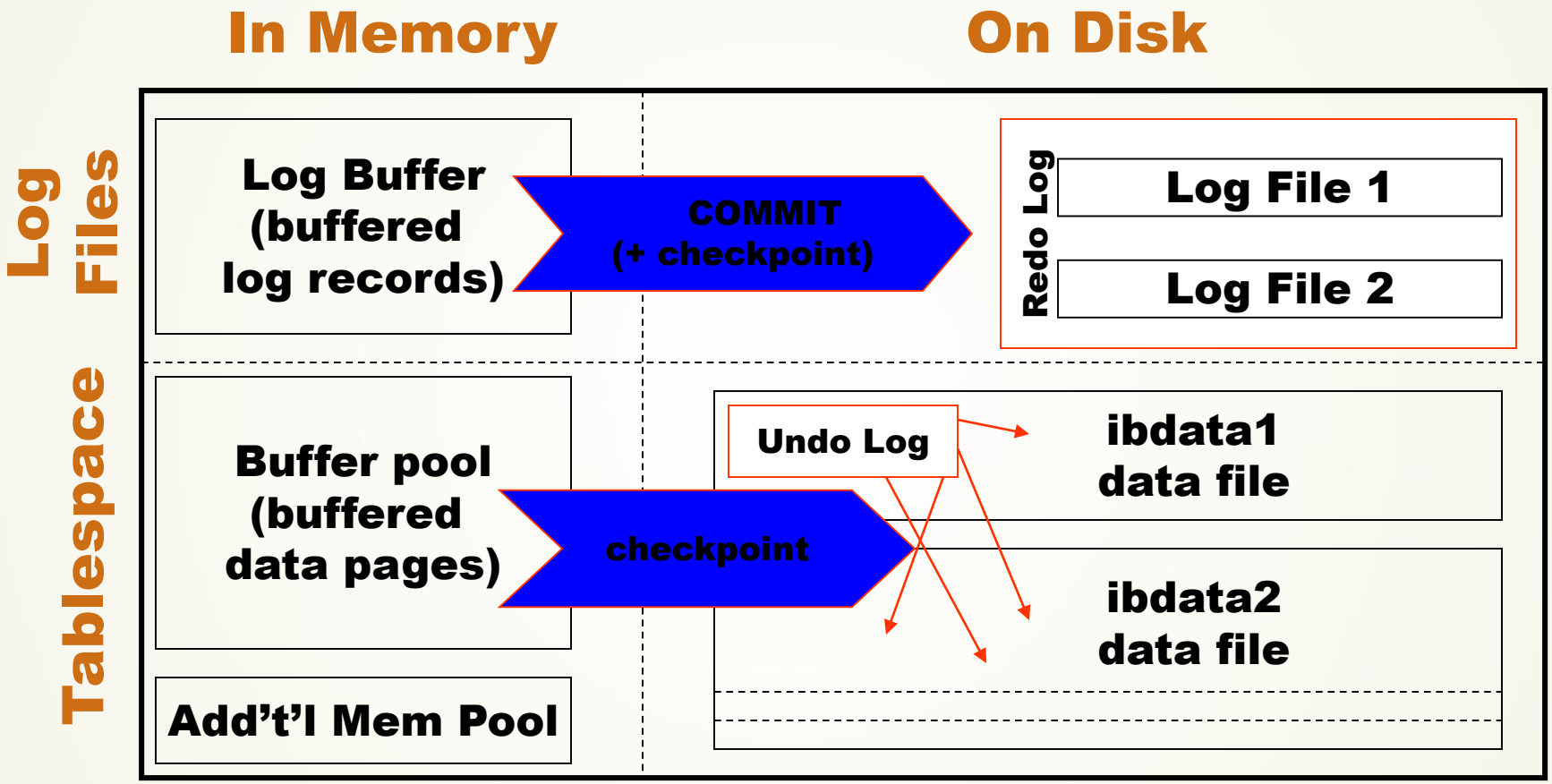
Memory Management

- Buffer pool: data pages; index pages; undo records; adaptive hash indexes; table of lock info
- Log buffer: redo records
- Additional memory pool: cached data dictionary; open table handles

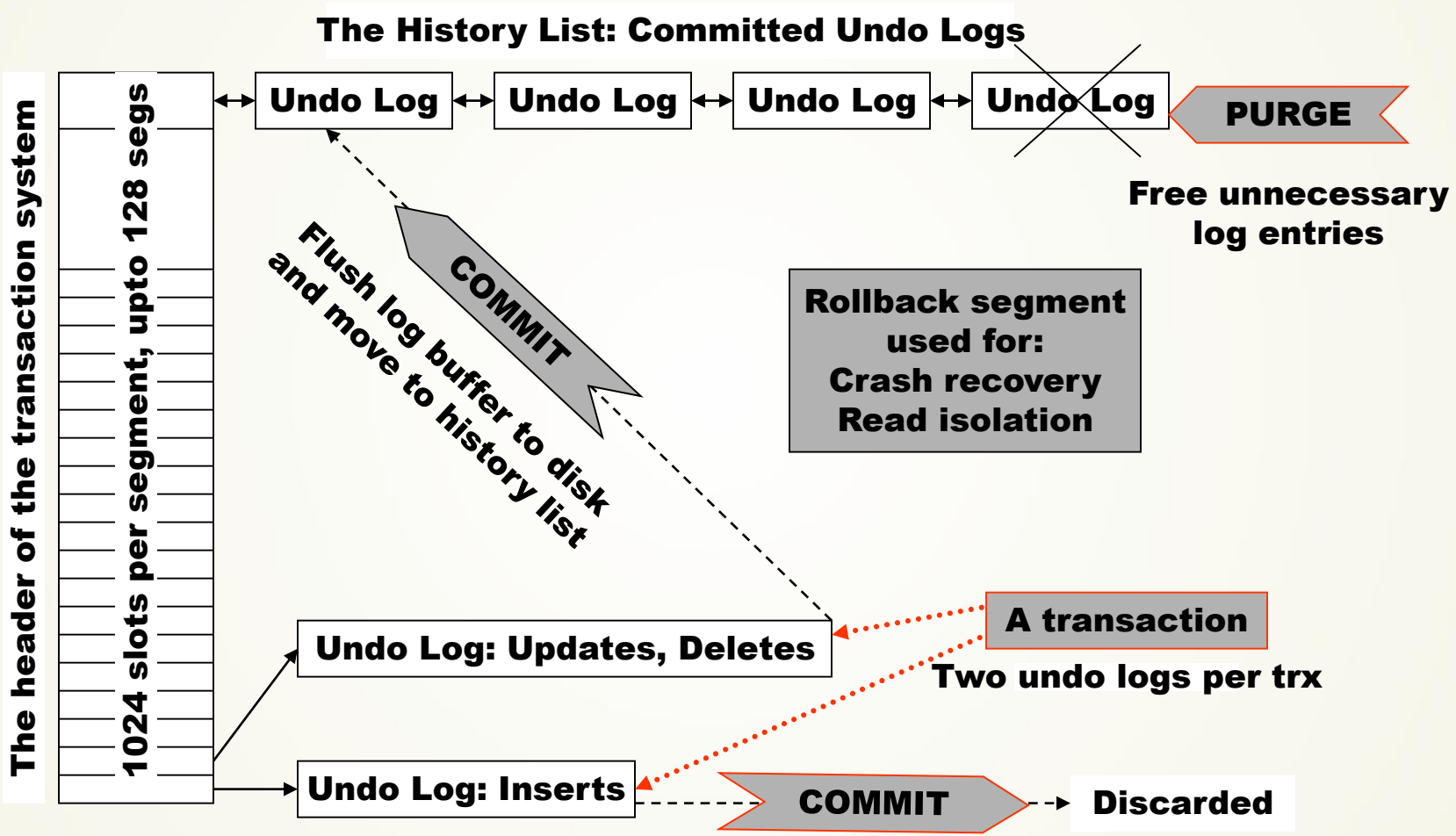
- Multiple buffer pool instances
- LRU, MRU

Threads

- User threads (MySQL server threads)
- Master thread
- IO threads
  - read io
  - write io
  - ibuf io
  - log io
- Purge threads
- Page cleaner (flush) thread
- Deadlock detection thread
- FTS, Statistics, Monitor, Drop table, Dump buffer pool, and more


In Memory
On Disk
Log Files
Tablespace
Log Buffer (buffered log records)
COMMIT (+ checkpoint)
Redo Log
Log File 1
Log File 2
Buffer pool (buffered data pages)
checkpoint
Undo Log
ibdata1 data file
ibdata2 data file
Add't'l Mem Pool

InnoDB Transaction Handling



Checkpointing

- A checkpoint is a log sequence number (LSN) such that: the data pages in the files contain all changes to the database earlier than LSN
- InnoDB's redo log files have a fixed capacity
- The 'age' of the latest checkpoint must not exceed this capacity
- If the checkpoint age would become too old, InnoDB writes the oldest

Flushing

- Activity of writing dirty pages & logs to the disk.
- There are two types of flushing:
  - **LRU flushing**, based on LRU_list (roughly ordered on time since last access)
  - **Adaptive flushing**, based on flush_list (strictly ordered on oldest_modification LSN)

Flushing (cont.)

- Flushing a batch typically involves:
  - Scanning the tail of the relevant list to find victims
  - Select neighbors as candidates for flushing as well
  - Copy dirty pages to the doublewrite buffer
  - Writing double write buffer to disk
  - Sync double write buffer
  - Write to data files
  - Sync all data files

Purging

- Purge is a type of garbage collection.
- Purge includes:
  - Remove obsolete values from indexes
  - Remove delete marked records that will not be seen by any active transaction
  - Remove the relevant undo entries from history list (a.k.a rollback segment)
- Multi threaded purge: perform purge on a periodic schedule

Change Buffer Merging

- Allows changes to secondary index leaf blocks to be deferred when block is not in the buffer.
- Three types of buffering: insert, delete, purge
- Merging
  - Choose random page from the buffer
  - Open a cursor on a random record on that page
  - Read buffer entries from that cursor to find at most 8 pages that should be fetched
  - Issue async IO requests; when async read IO completes, callback is done to apply deferred changes

Prefetching

- buf_read_ahead_random: before requesting a block read:
  - Count number of blocks in extent that were recently read based on position in buffer pool LRU list.
  - If more than 13 were recently read, prefetch others
- buf_read_ahead_linear: may be used when
  - Accessing first or last page in extent
  - Many pages (56) in extent have been accessed
  - Access pattern was sequential
  - When used, issue read requests for extent that contains that page that follows or precedes

Crash Recovery

- Commonly, there are 3 phases in Recovery

- There are 4 phases in InnoDB recovery
  - Recover incomplete pages from doublewrite buffer
  - Scan: read redo logs from disk and insert redo log entries into a red-black tree which is sorted on LSN
  - Redo: insert 'dirty' pages into the "flush_list"
  - Undo: incomplete transactions are rolled back

InnoDB Online Operations

Online Operations in MySQL 5.6

ADD PRIMARY KEY

ADD INDEX

DROP INDEX

ADD COLUMN

ADD FOREIGN KEY

DROP COLUMN

RENAME COLUMN

DROP FOREIGN KEY

RENAME TABLE

ALTER KEY_BLOCK_SIZE

ALTER COLUMN NULLABLE

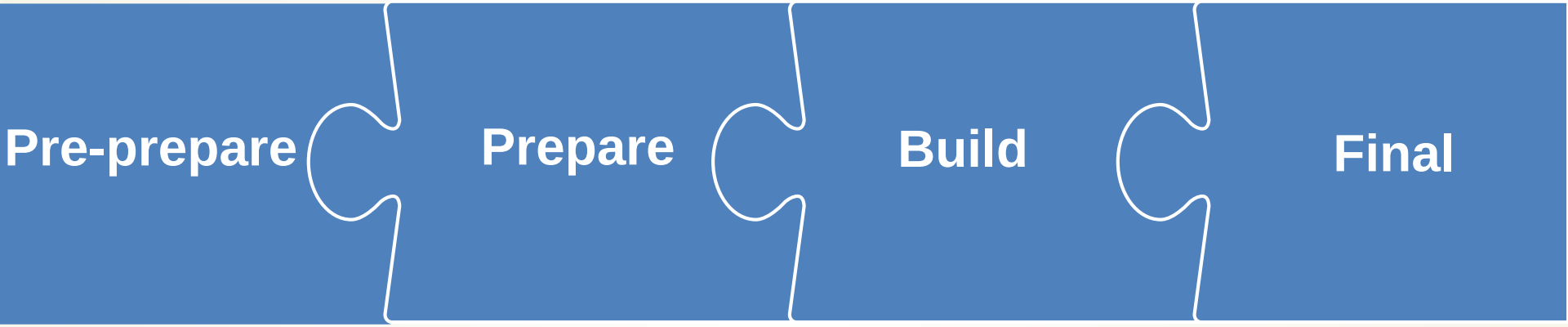
ALTER COLUMN NOT NULL

ALTER ROW FORMAT

Type of Online Operations

- Metadata only
  - MySQL Server metadata, such as alter column default
  - MySQL Server metadata & InnoDB metadata, such as add/drop foreign key
- Metadata plus w/o rebuilding the table, such as add/drop index
- Metadata plus rebuilding the table, such as add primary index, add column.

How Does It Work?



Pre-preparation Phase

- Server
  - Determine the algorithm and concurrency level supported by the storage engine.
  - Hold MDL_SHARED_UPGRADABLE: allow concurrent DML
- InnoDB

  ha_innobase::check_if_supported_inplace_alter()
  - Check if InnoDB supports a particular alter table in-place.

Prepare Phase

- Server
  - Upgrade to MDL_EXCLUSIVE: no concurrent DML allowed
  - Build internal objects describing requested changes
- InnoDB

  ha_innobase::prepare_inplace_alter_table()
  - Check whether the alter is legitimate
  - Update internal structures
  - Create temporary file(s) for change log(s) due to DMLs
  - Start logging

Build Phase

- Server
  - Hold MDL_SHARED_UPGRADABLE: allow concurrent DML
  - Let storage engine to carry out the changes requested by ALTER.
- InnoDB

  ha_innobase::inplace_alter_table()
  - Alter the table in-place with operations specified.
  - Apply the change logs

Final Phase

- Server
  - Update .frm and remove old table definitions
  - Upgrade to MDL_EXCLUSIVE: no concurrent DML allowed
  - Notify storage engine
  - Cleanup internal structures
- InnoDB

  ha_innobase:: commit_inplace_alter_table()
  - Commit or rollback the changes
    a) Sync and delete the logs
    b) Commit metadata changes
    c) Cleanup internal structures

Online Add Index

CREATE INDEX *index_name* ON *table name (column)*

	Concurrent User	Source (table)	(cluster) Index	Metadata Lock
Pre-Prepare Phase	Concurrent Select, Delete, Insert, Update	Check whether the online DDL is supported		Upgradable Shared Metadata Lock
Prepare Phase	No concurrent DML allowed	Create temp table for new index (if primary)	Create log files; Logging starts	Exclusive Metadata Lock
Build Phase	Concurrent Select, Delete, Insert, Update	Scan clustered index; Extract index entries; Sort / merge index build	DML Logging; Apply logs at the end of create index	Upgradable Shared Metadata Lock
Final Phase	No concurrent DML allowed	Drop old table (if primary)	Update system tables (metadata)	Exclusive Metadata Lock

InnoDB Monitoring & Diagnostics

Overview of Monitoring & Diagnostics

- SHOW ENGINE INNODB STATUS
- InnoDB Monitors
- Performance schema for InnoDB
- Information schema tables
  - Information schema metrics table
  - Information schema for InnoDB system tables
  - Information schema for InnoDB buffer pool

INNODB Show Status and Monitors

- Typical sections of monitor output
  - BACKGROUND THREAD
  - SEMAPHORES
  - LATEST FOREIGN KEY ERROR
  - LATEST DETECTED DEADLOCK
  - TRANSACTIONS
  - FILE I/O
  - LOG
  - BUFFER POOL AND MEMORY
  - ROW OPERATIONS

Performance Schema in InnoDB

- 46 mutexes
- 12 rwlocks
- 7 types of threads
- 3 types of I/O (data, log, tmpfile)
- More with debug binaries

Performance Schema in InnoDB

Types of running InnoDB threads from THREADS table

```
mysql> SELECT DISTINCT(name) FROM threads WHERE name LIKE "%innodb%";
+----------------------------------------+
| name                                   |
+----------------------------------------+
| thread/innodb/io_handler_thread        |
| thread/innodb/srv_lock_timeout_thread  |
| thread/innodb/srv_error_monitor_thread |
| thread/innodb/srv_monitor_thread       |
| thread/innodb/srv_master_thread        |
| thread/innodb/srv_purge_thread         |
| thread/innodb/page_cleaner_thread      |
+----------------------------------------+
7 rows in set (0.00 sec)
```

Information Schema Tables

29 INFORMATION_SCHEMA tables in InnoDB

- 9 Data Dictionary related
- 7 FTS related
- 6 Compression related
- 3 Buffer Pool related
- 3 on Locks / Transactions
- 1 General Statistics gold mine (metrics table)

Information Schema Metrics Table

17 modules, 207 counters

```
mysql> select DISTINCT subsystem from innodb_metrics order by subsystem;
+----------------------+
| subsystem            |
+----------------------+
| adaptive_hash_index |
| buffer               |
| buffer_page_io       |
| change_buffer        |
| compression          |
| ddl                  |
| dml                  |
| file_system          |
| icp                  |
| index                |
| lock                 |
| metadata             |
| os                   |
| purge                |
| recovery             |
| server               |
| transaction          |
+----------------------+
17 rows in set (0.02 sec)
```

```
mysql> select count(*) from
innodb_metrics;
+----------+
| count(*) |
+----------+
|      207 |
+----------+
1 row in set (0.00 sec)
```

InnoDB System Tables

```
mysql> SHOW TABLES LIKE 'INNODB_SYS_%';
+-----------------------------------------+
| Tables_in_information_schema (INNODB_SYS_%) |
+-----------------------------------------+
| INNODB_SYS_DATAFILES                    |
| INNODB_SYS_TABLESTATS                   |
| INNODB_SYS_INDEXES                      |
| INNODB_SYS_TABLES                       |
| INNODB_SYS_FIELDS                       |
| INNODB_SYS_TABLESPACES                  |
| INNODB_SYS_FOREIGN_COLS                 |
| INNODB_SYS_COLUMNS                      |
| INNODB_SYS_FOREIGN                      |
+-----------------------------------------+
9 rows in set (0.00 sec)
```

Get Involved!


Q&A
QUESTIONS
ANSWERS

欢迎莅临

2013中国数据库技术大会