InnoDB: What’s new in 8.0

Sunny Bains
Director Software Development
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

What Is New In 8.0
Future Directions
Q&A + Discussion
Integrated Data Dictionary

Legacy issues

• Up to 5.7 two separate data dictionaries (.frm & InnoDB DD)
• Changes were not atomic
• Mismatch between .frm files and InnoDB’s meta-data
• Concurrent access had to be very carefully managed
  • Separate locking/latching mechanisms
    • MDL, dict_sys_t::mutex, dict_sys_t::rw_lock etc.
  • Prone to deadlocks
• Not crash proof
Integrated Data Dictionary

Benefits

• One source of truth - Server meta-data
• Atomic DDL
  • No more .frm and InnoDB data dictionary mismatch issues
• Required for transactional DDL (future)
• Data dictionary tables stored in a transactional engine
• Control meta-data access using a single locking mechanism (MDL)
• Server supports the concept of Tablespaces
  • .frm files were per table, made general tablespace support messy
• .frm files not created for temporary tables - meta-data in memory only
Integrated Data Dictionary

Architecture
Performance

• Cost Based Optimiser statistics
  • Number of pages in RAM per index
• Remove the buffer pool mutex (Percona contribution)
  • Took a long time to fix problems in the contributed patch
    • Oracle QA team found lots of problems in edge cases - great effort
    • Foundation for more improvements in the future
• Redo log improvements (Alibaba contribution)
  • Improves redo scalability
Undo log changes

- Change the undo roll ptr format – upgrade impact
  - More flexible tablespace management
  - Implications for upgrade
- Default will be two undo tablespaces
- SQL syntax to manage undo logs dynamically
  - Exact syntax work in progress
  - `CREATE UNDO TABLESPACE 'rbs01' ADD DATAFILE 'rbs01.ibu';`
  - `DROP UNDO TABLESPACE 'rbs01';`
  - `ALTER UNDO TABLESPACE 'rbs01' SET OFFLINE/ONLINE;`
Performance (cont.)

- Group records by table id when purging
  - Reduces contention of the dict_index_t::lock when having multiple purge threads
- `--innodb_stats_include_delete_marked := bool`
  - Include/Exclude rows that are delete marked (in 8.0.1)
- `--innodb_deadlock_detect := bool (dynamic)`
  - On high concurrent loads, rely on `--innodb_lock_wait_timeout` and rollback
- Internal read ahead row buffer set by the Optimiser
Encryption and Generalised Tablespace Improvements

• Encryption of redo and undo log
  • --innodb-redo-log-encrypt := bool
  • --innodb-undo-log-encrypt := bool

• Generalised/Shared tablespaces
  • Support Encryption
  • Support Compression
  • Support Import/Export
Feature Improvements

• Memcache improvements
  • Support multiple get and range search
• Persistent auto increment
  • Doesn’t reset to SELECT MAX(AUTOINC_COL) FROM T; on restart
  • Probably the most requested feature since v3.x
• Bug 199 - Created on 27 March 2003
Information Schema

- A new INFORMATION_SCHEMA table, `INNODB_CACHED_INDEXES`
- Report pages cached in the InnoDB buffer pool for each index.
Descending Indexes

- Change buffering is not supported
  - If secondary index contains a descending index key column
  - If the primary key includes a descending index column
- Supported for all data types for which ascending indexes are available.
- Supported for ordinary and generated columns (both VIRTUAL and STORED)
- Not supported for full text indexes and RTree
NO_WAIT/SKIP LOCKED

• If NO_WAIT set for a query
  • Return immediately without waiting for the row lock to be released
  • SELECT * FROM T WHERE C1 = n and C2 = m FOR UPDATE NO_WAIT;

• If SKIP LOCKED set for a query
  • Skip locked row, without waiting for the row lock to be released
  • SELECT * FROM T WHERE C1 = N AND C2 = m LIMIT 1 FOR UPDATE SKIP LOCKED;
Refactoring

• C++11
• Created libinnodb_zipdecompress.a
  – Isolated code to write tools
• Now uses the C++ std::thread library for thread management
• BLOB code refactored to provide an internal C++ interface on compressed/uncompressed BLOB data
• The InnoDB start up code was refactored.
  • Cleaner control flow
Miscellaneous

• Avoid intermediate commits that would occur every 10000 rows
  • e.g. ALTER TABLE ... ALGORITHM=COPY
• Remove .isl files (InnoDB Symbolic Link files)
  • Used when creating tablespace data files outside of the MySQL data directory.
  • Redo log records are now used to locate all tablespaces
• --innodb-scan-directories := “dir1;dir2;...dirN”;
• --innodb-read-only semantics change
  • If ON then affects entire instance
• Lots of Bug fixes
Deprecations / Removals

• Deprecated parameters that have been removed
  • innodb_file_format
  • innodb_file_format_check
  • innodb_file_format_max
  • innodb_large_prefix
  • innodb_stats_sample_pages
  • innodb_locks_unsafe_for_binlog
  • innodb_checksums
  • innodb_support_xa (always ON)
  • Innodb_undo_logs
Future Directions
Better Tablespace Management

• Versioning for tablespaces
  • Support multiple tablespace/page/row formats simultaneously
  • Easier to introduce new features/capabilities
    • E.g., a page/row format that gives better compression
  • Support the Server native row format, avoid conversions
  • Improve upgrade process
  • Helps with maintaining backward compatibility
Better Tablespace Management

- SQL for managing UNDO logs/tablespaces
- Tablespaces will be self describing:
  - Serialized Dictionary Information (SDI)
  - Embedded inside the table space (Current DMR they are separate for non-InnoDB)
  - InnoDB keeps two copies
  - Improve import/export - long term objective is to make it instant
- Getting rid of the legacy “system tablespace” a.k.a ibdata
Serialized Dictionary Information (SDI)

```json
{
    "sdi_version": 1,
    "dd_version": 1,
    "dd_object_type": "Table",
    "dd_object": {
        "name": "tbl1",
        "mysql_version_id": 80000,
        "created": 20160922042352,
        "last_altered": 20160922042352,
        ...
        "columns": [
            {
                "name": "id",
                "type": 4,
                "is_nullable": false,
                ...
            },
            ...
        ],
        "indexes": [
            ...
        ],
        "foreign_keys": [],
        "partitions": [],
        "collation_id": 8
    }
}
```
Memory Storage Engine

• Currently for internal use only (Optimizer joins etc.)
• Not shared across connection
• Lifetime limited to query life time
• Limited size, bounded by memory allocated
• Less space overhead per row compared to current memory engine
Support Document Store And JSON

• More flexible BLOB handling
  • Allow partial fetch and update
  • Plan is to make streaming easier
Improved Tooling

- A tool for extracting Serialized Dictionary Information (SDI)
  - ibd2sdi
  - Works offline and online
  - Extracts the SDI record id, type, data in JSON format
  - Useful during disaster recovery
    - e.g., Table corrupted in a tablespace with multiple tables
    - Extract the meta-data from the .ibd file into a separate .SDI file
    - Remove corrupt table meta-data by editing .SDI file
    - Use edited .SDI file to import the tablespace and ignore the corrupted table
Improved Tooling

• For 5.7 instances
• Move undo logs out of the system tablespace
• Change number of rollback segments per UNDO tablespace
• Truncate the system tablespace up to last used extent
Dedicated_Server := boolean (Default true)

• Out of the box our defaults are now much better
• Exception being resource usage where we assume 512M RAM
  • Is it a developer laptop or a production server?
  • Same installation packages
  • Requirement that installers can not ask questions
• Dedicated Server scales primary resources according to system shape:
  • InnoDB Buffer Pool and Log File Size
• Friendly for Cloud and VM environments
• Advanced users will still have the freedom to make configuration changes
  • Beginners will no longer have to
Flush log at transaction commit configuration changes

- Innodb-flush-log-at-trx-commit := 0 | 1 | 2
  - 0 – write and flush once per second
  - 1 – write and flush at each commit
  - 2 – write at commit, flush once a second
- On modern HW once a second can be 1000s of lost transactions
  - Keep above semantics
  - Setting to a number N > 2 will flush every N transaction commits
Performance

Latch free transaction life cycle
Latch free MVCC
Improved deadlock checking
Parallel threads for redo log writes (lock free)
Fix the file system mutex contention
Performance
Performance

- InnoDB Top-7 Mutex Spins/sec: Update-NoKEY 10Mx8 8.512usr MySQL-5.7/8.0/8.0-dev1 1S/2S@44cores-HT - [spin_rounds/s]
  - Graph showing mutex spins per second with various labels for different categories.

- InnoDB Top-7 Spin Waits/sec: Update-NoKEY 10Mx8 8.512usr MySQL-5.7/8.0/8.0-dev1 1S/2S@44cores-HT - [spin_waits/s]
  - Graph showing spin waits per second with various labels for different categories.