InnoDB in 10.2, 10.3 and beyond
Some MariaDB changes and plans
Short history of InnoDB and Marko Mäkelä

1995 Created by Heikki Tuuri (who shortly worked at Solid)
2002 InnoDB included in MySQL 3.23
2003 Marko Mäkelä joins as the first full-time employee
2005 Heikki Tuuri sells Innobase Oy to Oracle Corporation
2010 Oracle acquires Sun Microsystems (which had acquired MySQL)
2016 Marko Mäkelä joins MariaDB as Lead Developer InnoDB
MySQL 5.7 InnoDB Features Not In MariaDB

- Oracle version of encryption, page compression
  - MariaDB 10.1 already shipped something similar
- MDEV-11816 Disallow CREATE TEMPORARY TABLE...COMPRESSED
- MDEV-12050 Remove unused InnoDB Memcached hooks
  - MySQL 5.6+ ships a patched snapshot of an abandoned Memcached dev branch
  - InnoDB Memcached was never enabled in MariaDB (dead code in 10.0, 10.1)
- CREATE TABLESPACE ...; CREATE TABLE ... TABLESPACE=...
  - Tablespaces are not proper ‘native’ objects (even in MySQL, no import/export)
  - MDEV-11426 Remove InnoDB INFORMATION_SCHEMA.FILES implementation
- MDEV-11487 Revert InnoDB internal temporary tables from WL#7682
  - In MariaDB, query execution uses Aria tables; even MyISAM works just fine
- InnoDB native partitioning (ha_innopart)
  - Memory usage can be reduced by removing the row_prebuilt_t prefetch buffer
New InnoDB features in MariaDB 10.2

- **MDEV-6076** Persistent AUTO_INCREMENT
  - AliSQL idea: Repurpose PAGE_MAX_TRX_ID on the clustered index root page
  - Update atomically in the DML mini-transaction (before transaction commit)
  - Differences from MySQL 8.0.0:
    - No redo log format change
    - No dependency added to data dictionary; preserved on export/import

- **MDEV-11824** Allow ROW_FORMAT=DYNAMIC in the system tables

- **MDEV-12289** Keep 128 persistent rollback segments for compatibility and performance
  - Do not reserve 32 persistent rollback segments for temporary tables.
  - Unlike MySQL 5.7, allow an upgrade without prior innodb_fast_shutdown=0.
New InnoDB features in MariaDB 10.2

• **MDEV-12219** Discard temporary undo logs at transaction commit
  – There is no purge or MVCC for TEMPORARY TABLE
• Cleaned up startup and shutdown (no memory leaks after failed startup)
  – Some refactoring of the undo log, purge, and transaction system code
  – Removed some race conditions around events and thread status variables
  – Added regression tests
  – **MDEV-11915** Detect InnoDB system tablespace size mismatch early
  – **MDEV-11947** InnoDB purge workers fail to shut down
  – **MDEV-11985** Make innodb_read_only shutdown more robust
• **MDEV-11520** (5.5+) Properly use posix_fallocate() for extending files
  – 10.1+ page_compression: Do not physically preallocate sparse files
InnoDB features in MariaDB 10.3

- **MDEV-10139** Support for `SEQUENCE` objects (MariaDB 10.3.1)
  - Implemented as a no-rollback single-index table (containing a single record)
  - No-rollback tables could find other uses, such as a table that stores cross-engine transaction state (XID, GTID, start/end id, timestamps) (**MDEV-11657**)
  - (2 days of InnoDB work; many days of SQL layer work by Monty)

- **MDEV-11369** Instant `ADD COLUMN` (MariaDB 10.3.2)
  - Compatible with existing formats (except `ROW_FORMAT=COMPRESSED`)
  - No data dictionary format change; all metadata is stored in the clustered index
  - Compatible with export/import
  - In 10.4, **MDEV-11424** would allow any `ALTER TABLE` that does not involve data conversions that could fail, or building any indexes
InnoDB Crash Recovery Changes

- **MDEV-11027** (10.0+) Better progress reporting for InnoDB crash recovery
- **MDEV-11556** (10.1+) InnoDB redo log apply fails to adjust data file sizes
  - Redo log scan tracks FSP_SIZE changes; files are extended when first opened
  - This bug was always worked around by InnoDB Hot Backup (innobackup) and later MySQL Enterprise Backup (MEB), and also Percona XtraBackup.
    - They would silently extend the file on seemingly out-of-bounds page number.
- **MDEV-11690** (10.2) Remove UNIV_HOTBACKUP (used by MEB only)
- **MDEV-11782** (10.2) Redefine the innodb_encrypt_log format
  - Implement proper upgrade when using redo log encryption.
  - Always rebuild the redo log when disabling or enabling encryption.
- **MDEV-11814** Refuse innodb_read_only startup if crash recovery is needed
- **MDEV-12061** (10.2) Allow innodb_log_files_in_group=1
- **MDEV-12103** (10.2) Reduce the time of looking for MLOG_CHECKPOINT
InnoDB performance improvements

- **MDEV-12121** (10.2) Introduce build option WITH_INNODB_AHI=OFF
  - Could benefit users of innodb_adaptive_hash_index=0.
  - The InnoDB adaptive hash index can improve performance for some workloads
  - It could also hurt performance. And it incurs an overhead for DDL operations.

- **MDEV-12288** (10.3.2) Reset DB_TRX_ID when the history is removed
  - Changes the undo log format!
  - Changes the B-tree format (by allowing “null pointers” in DB_TRX_ID)
  - Prepares for **MDEV-11658** (faster IMPORT of .ibd files)

- **MDEV-11585** (10.2) Hard-code the shared InnoDB temporary tablespace ID
  - MySQL 5.7 would reassign the built-in temporary tablespace ID on every startup.
  - Simpler, faster code and easier troubleshooting with a compile-time constant.
10.4: Bulk-Loading for InnoDB

• Shaohua Wang implemented WL#7277 for `ALTER TABLE...ALGORITHM=INPLACE` in MySQL 5.7, to speed up `ADD INDEX` and table rebuilds
• Extend this to inserts into an empty table or partition, while disabling low-level logging
  – `ALTER TABLE...ALGORITHM=COPY`, `CREATE TABLE...SELECT`
  – `LOAD DATA INFILE`, `INSERT`, anything!
• Recovery: Write a transaction log record that causes `TRUNCATE` or `DROP`
• Optionally write redo log for physical replication (like Aurora, PolarDB)
Vision: Making InnoDB truly “files-per-table”

• Global persistent state should be reduced to the minimum:
  – Corruption of global state can make the whole server unavailable!
  – InnoDB data dictionary, undo logs and change buffer make it harder to transfer files between instances.

• Bare minimum global state: Transaction state (was it committed or not?)
  – Introduce *.ibu files for per-table or per-partition transaction (undo) logs.
  – Remove the InnoDB data dictionary; rely on the server dictionary.
    • Keep *.frm files; store FOREIGN KEY constraints in new *.frk files.
    • Make *.ibd files self-contained (store index root page numbers and FULLTEXT indexes)
  – Introduce a transactional ddl_fixup table to make DDL operations transactional
    • Rollback or roll-forward “DROP” operations; roll-back “RENAME” operations.

• This will allow instant IMPORT (no need to rewrite the file)
  – If there is old history (* .ibu file), it can be removed efficiently.
Vision: Cross-Engine BACKUP in the Server

- **BACKUP** \langle tablenames \rangle \ [FROM \langle time \rangle] \ [TO \langle directory \rangle]
- **Like** FLUSH TABLES \langle tablenames \rangle **FOR** EXPORT, **but supports:**
  - Streaming in query result format (without TO \langle directory \rangle):
    - \langle WRITE, filename, offset, length, data \rangle
    - \langle DELETE, filename, 0, 0, NULL \rangle
    - \langle RENAME, filename, 0, 0, newfilename \rangle
  - With FROM \langle time \rangle, this can be a differential backup
    - Full backup, unless the storage engines have the history since the last backup
    - The \langle time \rangle could be a cross-engine LSN (cross-engine write-ahead log).
- No log needs to be shipped; just ready-to-use data files.
- No separate “restore” tool is needed: just let the server discover the files.
  - Partial restore may still require an intermediate server.
Vision: Cross-Engine Native Transactions

• One engine will not fit all: Mixed operation with MyRocks, ColumnStore, ...
  – MariaDB should be a collection of transactional engines that play well together

• Make InnoDB “dumber” and support the features for all storage engines:
  – (10.3 supports compressed columns in the SQL layer, not just for InnoDB)
  – Global transaction state (including XID, GTID, timestamps for temporal tables)
  – Global write-ahead log (write to a single log to mark transaction committed)
  – Move FOREIGN KEY processing to the SQL optimizer/executor
  – Move FULLTEXT INDEX implementation to the SQL layer

• Maybe even: Allow the indexes of a table to exist in different engines
  – InnoDB could support SPATIAL INDEX for MyRocks tables
  – Implementation: by a more advanced “partitioning” engine?
Questions?

Thank you for your attention!
marko.makela@mariadb.com