STORAGE ENGINES

MyISAM, InnoDB, Archive, Memory

STORAGE ENGINES

- As discussed throughout the course, MySQL tables are all backed to a pluggable storage engine.
- The storage engine defines the features available to the table, as well as the underlying storage and organization of the data.
- There are a number of engines available for use with MySQL, which will be explored in the following slides.

MYISAM

- The MyISAM storage engine was the original engine for all MySQL systems from 3.23 until 5.5, when it became InnoDB.

- MyISAM is a direct descendent of the ancient ISAM table format (deprecated), with a focus on speed.

- MyISAM is non-transactional.

- Transactional? What now?

TO TRANSACTION OR NOT TO TRANSACTION

- **Transactional**
  - Safer from server crash
  - Run multiple changes in a single transaction
  - Rollback your changes if something goes wrong
  - Better write concurrence
- **Non-Transactional**
  - Faster – No transaction overhead
  - Smaller file system footprint
  - Smaller RAM requirements

MYISAM

- Each table is composed of 3 files
  - `table.frm` Format File: Stores table definition
  - `table.MYD` Data File: Stores the data
  - `table.MYI` Index File: Stores index data
- Table files can be moved to another database/server by copying. Should only be performed under set conditions as discussed in backups lecture.
- Supports full text searching and extensive indexing capabilities
- No deadlocks – Uses full table lock for Inserts - at the cost of reduced concurrency.

MYISAM

- Three possible table storage formats:
  - **Fixed**
    - All rows the same size - fast lookups, but takes more space on disk
  - **Variable**
    - Not all rows the same size, so takes less space to store
  - **Compressed**
    - Packed to save space, fast retrieval, **read-only**

MYISAM PROS

- Supports FULLTEXT/Spatial Indexes
- Low filesystem foot prints
- Uses less RAM
- Fast SELECT/INSERT (append) performance
- Maintains an internal row count (COUNT(*) is fast)

MYISAM CONS

- No transactional support
- Table level locking only
- No crash recovery
- Blocking backups
- No support for Foreign Keys

INNODB

- Transactional Engine - Supports COMMIT, SAVEPOINT, ROLLBACK
- By default data/index stored in `$DATADIR/ib_data*` files.
- Like all MySQL Storage engines, there will be a table format file (`.frm`)
- By default the tablespace is shared for all database/tables in server. But tables can be configured to use individual tablespaces (see 13.2.2.1).
  - NOTE: Even with this option the default tablespace `/var/lib/mysql/ibdata` is needed. NEVER delete an `ib_data` file if unsure of what you're doing.
- InnoDB logs are used to store transaction activity. Can be deleted/resized if MySQL is stopped properly.

INNODB

- Full Atomicity, Consistency, Isolation, Durability or **ACID** compliance, for more on ACID go to:
  - `http://en.wikipedia.org/wiki/ACID`
- Auto recovery in case of server crash (Durability)
- Multi-versioning concurrency control (MVCC)
- Row-level locking - Great for WRITE concurrency
- Supports Foreign Keys!

INNODB PROS

- ACID Compliance
- Crash auto recovery
- High Storage Limit (64TB per tablespace!)
- Row-level locking
- Foreign Keys
- MVCC Support
- Clustered Indexes
- Non-blocking online backups

INNODB CONS

- No Fulltext/Spatial indexes
- Higher filesystem footprint (+2x)
- RAM hungry. Performance depends on having large buffers.

MEMORY

- Data and indexes are stored in RAM! Super performance!
- The table has a .frm file on disk. A restart will maintain the table structure but the data/index will be gone.
- Limited by max_heap_table_size setting (default 16M)
- Table level locking
- Cannot contain TEXT or BLOB.
  - NOTE: This is one of the reason why lots of temporary tables can be created on disk when we use a JOIN on tables containing TEXT or BLOB!

MEMORY PROS

- Fast reads/writes
- Support for Hash and Tree indexes

MEMORY CONS

- Volatile data
- Limited size
- No transactions
- Table level locking
- No support for foreign keys
- No TEXT/BLOB fields

ARCHIVE

- Stores large amounts of data without indexes in a very small footprint
- Supports `INSERT` and `SELECT`, but not `DELETE`, `REPLACE`, or `UPDATE`
- Table composed of three files:
  - `.frm` format file
  - `.ARZ` data file
  - `.ARM` meta data
- Uses row level locking
- Rows are compressed on insertion and uncompressed on retrieval
- Designed for efficient archival storage of large amounts of data
- data warehouse applications, data archiving, data auditing

ENGINES

- To view the available engines on a server:
  - SHOW ENGINES;
- To view the engine a table is using:
  - `SHOW CREATE TABLE db.table`
  - `SHOW TABLE STATUS`
  - `SHOW TABLE STATUS LIKE 'db.table'`

USING ENGINES

- To create a table with a non-default storage engine:
  - `CREATE TABLE table ( ... ) ENGINE=engine`
- Example:
  - ```
    CREATE TABLE test (
      testcol INT
    ) ENGINE=InnoDB;
    ```
    ```

CHANGING ENGINES

- It is possible to change the engine of an existing table using the `ALTER TABLE` statement:
  - `ALTER TABLE test ENGINE=MyISAM;`
- Will increase CPU load and I/O latency while running
- Conversion process may fail if:
  - The target engine doesn't support all features used in the original engine.
  - The table data exceeds the capabilities of the new engine.
- In the event of a conversion failure, MySQL will simply continue using the original table engine.

LAB

1) Convert the movie table to InnoDB.

2) Bonus: Refer to sections 13.2.3 and 13.2.5 in the MySQL Reference Manual and carefully add an additional 100M auto-extending tablespace for InnoDB.

3) Consider the possible performance impact of separating out tablespaces as well as index files for InnoDB across multiple disks.

```
slideshow.end();
```