- **InnoDB enhancements.** These `InnoDB` enhancements were added:

  - The current maximum auto-increment counter value is written to the redo log each time the value changes, and saved to an engine-private system table on each checkpoint. These changes make the current maximum auto-increment counter value persistent across server restarts. Additionally:

    - A server restart no longer cancels the effect of the `AUTO_INCREMENT = N` table option. If you initialize the auto-increment counter to a specific value, or if you alter the auto-increment counter value to a larger value, the new value is persisted across server restarts.

    - A server restart immediately following a `ROLLBACK` operation no longer results in the reuse of auto-increment values that were allocated to the rolled-back transaction.

    - If you modify an `AUTO_INCREMENT` column value to a value larger than the current maximum auto-increment value (in an `UPDATE` operation, for example), the new value is persisted, and subsequent `INSERT` operations allocate auto-increment values starting from the new, larger value.

    For more information, see Section 15.6.1.6, “AUTO_INCREMENT Handling in InnoDB”, and InnoDB AUTO_INCREMENT Counter Initialization.

  - When encountering index tree corruption, `InnoDB` writes a corruption flag to the redo log, which makes the corruption flag crash safe. `InnoDB` also writes in-memory corruption flag data to an engine-private system table on each checkpoint. During recovery, `InnoDB` reads corruption flags from both locations and merges results before marking in-memory table and index objects as corrupt.

  - The `InnoDB memcached` plugin supports multiple `get` operations (fetching multiple key-value pairs in a single `memcached` query) and range queries. See Section 15.20.4, “InnoDB memcached Multiple get and Range Query Support”.

  - A new dynamic variable, `innodb_deadlock_detect`, may be used to disable deadlock detection. On high concurrency systems, deadlock detection can cause a slowdown when numerous threads wait for the same lock. At times, it may be more efficient to disable deadlock detection and rely on the `innodb_lock_wait_timeout` setting for transaction rollback when a deadlock occurs.

  - The new `INFORMATION_SCHEMA.INNODB_CACHED_INDEXES` table reports the number of index pages cached in the `InnoDB` buffer pool for each index.

  - `InnoDB` temporary tables are now created in the shared temporary tablespace, `ibtmp1`.

  - The `InnoDB` tablespace encryption feature supports encryption of redo log and undo log data. See Redo Log Encryption, and Undo Log Encryption.

  - `InnoDB` supports `NOWAIT` and `SKIP LOCKED` options with `SELECT ... FOR SHARE` and `SELECT ... FOR UPDATE` locking read statements. `NOWAIT` causes the statement to return immediately if a requested row is locked by another transaction. `SKIP LOCKED` removes locked rows from the result set. See Locking Read Concurrency with NOWAIT and SKIP LOCKED.

    `SELECT ... FOR SHARE` replaces `SELECT ... LOCK IN SHARE MODE`, but `LOCK IN SHARE MODE` remains available for backward compatibility. The statements are equivalent. However,