DevSheets

MySQL Advanced

Types, JSON, indexes and query plans in 8.4.

01 Types & size limits

DECIMAL(10,2)	Exact decimal, for money
VARCHAR(255)	Up to 255 characters
TEXT	Up to 65,535 bytes
LONGTEXT	Up to 4 GB (MEDIUMTEXT 16 MB)
TIMESTAMP	UTC, 1970 to 2038-01-19
DATETIME	Years 1000 to 9999

Limits are bytes: utf8mb4 uses up to 4 per character.

02 Table options & engines

```
CREATE TABLE products (
  id BIGINT AUTO_INCREMENT PRIMARY KEY,
  status ENUM('active', 'pending'),
  notes TEXT DEFAULT (''),
  updated DATETIME DEFAULT CURRENT_TIMESTAMP
    ON UPDATE CURRENT_TIMESTAMP
) ENGINE=InnoDB;
```

TEXT, BLOB and JSON defaults need parentheses. InnoDB is the default engine; MEMORY tables empty on restart and reject TEXT.

03 JSON columns

```
CREATE TABLE events (
  id INT AUTO_INCREMENT PRIMARY KEY,
  data JSON NOT NULL
);
INSERT INTO events (data) VALUES
  ('{"user": "john", "action": "login"}'),
  (JSON_OBJECT('user', 'jane'));
```

Invalid JSON is rejected on insert.

04 Query & update JSON

```
SELECT data->>'$.user' FROM events
WHERE data->>'$.action' = 'login'
  AND JSON_CONTAINS(data, '"book"',
                    '$.items');
UPDATE events SET data =
  JSON_SET(data, '$.status', 'processed')
WHERE id = 1;
```

-> returns JSON ("john"); ->> unquotes it (john). Paths start at $; $.items[0] is the first element.

05 Index a JSON field

```
ALTER TABLE events
  ADD COLUMN username VARCHAR(255)
    GENERATED ALWAYS AS (data->>'$.user'),
  ADD INDEX idx_username (username);
SELECT id FROM events
WHERE username = 'jane';
```

JSON columns cannot be indexed directly. Generated columns are VIRTUAL by default; InnoDB can index them.

06 Index types

```
CREATE UNIQUE INDEX uk_email
  ON users (email);
CREATE INDEX idx_name
  ON users (last_name, first_name);
CREATE INDEX idx_email10
  ON users (email(10));  -- prefix
CREATE FULLTEXT INDEX ft_body
  ON articles (title, content);
SHOW INDEX FROM users;
```

Column order matters; every index slows writes.

07 Invisible, DESC & functional

```
ALTER TABLE users
  ALTER INDEX idx_name INVISIBLE;
CREATE INDEX idx_created
  ON orders (created_at DESC);
CREATE INDEX idx_month
  ON sales ((MONTH(sale_date)));
```

The optimizer ignores an invisible index but writes still maintain it: test a drop safely. Expressions need their own parentheses.

08 EXPLAIN & optimizer hints

```
EXPLAIN SELECT * FROM users WHERE id = 1;
EXPLAIN FORMAT=JSON SELECT * FROM orders
WHERE status = 'pending';
SELECT /*+ INDEX(users uk_email) */ *
FROM users WHERE email = 'jo@x.com';
ANALYZE TABLE users;  -- fresh statistics
```

Watch for "Using filesort" or "Using temporary". INDEX and NO_INDEX hints supersede FORCE and IGNORE INDEX.

09 Slow & running queries

```
SET GLOBAL slow_query_log = ON;
SET GLOBAL long_query_time = 2;
SELECT digest_text, count_star
FROM performance_schema
  .events_statements_summary_by_digest
ORDER BY sum_timer_wait DESC LIMIT 10;
SHOW STATUS LIKE 'Threads%';
SHOW FULL PROCESSLIST;
KILL QUERY 1234;  -- keeps the connection
```

SHOW PROFILE is deprecated; 8.0 removed the query cache.

MySQL Advanced

Transactions, stored code and operations.

10 Partitioning

```
CREATE TABLE orders (id INT, created DATE)
PARTITION BY RANGE (YEAR(created)) (
  PARTITION p24 VALUES LESS THAN (2025),
  PARTITION pmax VALUES LESS THAN MAXVALUE);
ALTER TABLE orders REORGANIZE PARTITION pmax
INTO (PARTITION p25 VALUES LESS THAN (2026),
  PARTITION pmax VALUES LESS THAN MAXVALUE);
```

ADD PARTITION cannot follow MAXVALUE, so split pmax. Strings need LIST COLUMNS. Unique keys must include the partitioning columns; no foreign keys.

11 Transactions & isolation

```
START TRANSACTION;
UPDATE users SET credits = 100;
SAVEPOINT step1;
DELETE FROM logs;
ROLLBACK TO step1;  -- undo the DELETE only
COMMIT;
SET SESSION TRANSACTION ISOLATION LEVEL
  READ COMMITTED;
```

InnoDB defaults to REPEATABLE READ.

12 Locking reads

```
SELECT * FROM orders WHERE id = 123
FOR UPDATE;            -- exclusive lock
SELECT * FROM products WHERE id = 7
FOR SHARE;             -- shared lock
SELECT * FROM jobs WHERE status = 'pending'
LIMIT 10 FOR UPDATE SKIP LOCKED;
```

Locks last until COMMIT; plain SELECTs are not blocked. NOWAIT errors at once. data_lock_waits shows blockers.

13 Stored procedures

```
DELIMITER $$
CREATE PROCEDURE GetUserCount(OUT total INT)
BEGIN
  SELECT COUNT(*) INTO total FROM users;
END$$
DELIMITER ;
CALL GetUserCount(@count); SELECT @count;
```

DELIMITER $$ lets the body keep its ; (mysql client only).

14 Errors & SIGNAL

```
DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN ROLLBACK; RESIGNAL; END;
START TRANSACTION;
UPDATE accounts SET balance = balance - amt
WHERE id = acct AND balance >= amt;
IF ROW_COUNT() = 0 THEN
  SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Insufficient funds';
END IF;
COMMIT;
```

In a procedure body: any error rolls back and is re-raised.

15 Functions & triggers

```
CREATE FUNCTION tax(price DECIMAL(10,2))
RETURNS DECIMAL(10,2) DETERMINISTIC
RETURN price * 0.08;
CREATE TRIGGER update_modified
BEFORE UPDATE ON users FOR EACH ROW
SET NEW.modified_at = NOW();
```

With binary logging on (the default), functions must declare DETERMINISTIC, NO SQL or READS SQL DATA.

16 Replication

```
-- on the source
CREATE USER repl IDENTIFIED BY 'pw';
GRANT REPLICATION SLAVE ON *.* TO repl;
SHOW BINARY LOG STATUS;  -- File, Position
-- on the replica
CHANGE REPLICATION SOURCE TO
  SOURCE_HOST='db1', SOURCE_USER='repl',
  SOURCE_PASSWORD='pw', SOURCE_LOG_POS=158,
  SOURCE_LOG_FILE='binlog.000001',
  GET_SOURCE_PUBLIC_KEY=1;
START REPLICA;
SHOW REPLICA STATUS\G
```

Give each replica its own server-id. caching_sha2_password (the default) needs GET_SOURCE_PUBLIC_KEY=1 or TLS.

17 Backup & restore

```
mysqldump -u root -p --single-transaction \
  --routines --events shop > shop.sql
mysql -u root -p shop < shop.sql
mysqlbinlog binlog.000001 binlog.000002 \
  | mysql -u root -p  # point-in-time replay
```

--single-transaction gives a consistent InnoDB dump without locks. --routines and --events add stored code.

18 Users & privileges

```
CREATE USER 'app'@'%'
  IDENTIFIED BY 'secure_password';
GRANT SELECT, INSERT, UPDATE
  ON shop.* TO 'app'@'%';
SHOW GRANTS FOR 'app'@'%';
```

GRANT no longer creates accounts: CREATE USER first.