

[TOC](#) [Previous](#) [Next](#)

MySQL Command Syntax

[Structured Query Language](#) [MySQL and SQL](#) [MySQL Identifiers](#) [MySQL Operators](#) [Comments](#)

Connections and sessions

[SET](#) [USE](#)

Data Definition Language

[CREATE DATABASE](#) [ALTER DATABASE](#) [RENAME DATABASE](#) [DROP DATABASE](#)
[CREATE | ALTER | DROP SERVER](#) [BACKUP DATABASE](#) [RESTORE](#)

[CREATE TABLE](#)

[CREATE definitions](#) [Column defs](#) [Silent column changes](#) [Keys](#) [PARTITION](#) [CREATE...SELECT](#)

[ALTER TABLE](#) [DROP TABLE](#) [RENAME TABLE](#) [CREATE | ALTER TABLESPACE](#)

[CREATE | ALTER LOGFILE GROUP](#) [CREATE | ALTER | DROP VIEW](#)

[CREATE INDEX](#) [DROP INDEX](#) [CREATE | ALTER | DROP EVENT](#)

[CREATE | DROP FUNCTION](#) [CREATE | DROP TRIGGER](#)

Database Administration Commands

[SHOW](#) [INFORMATION SCHEMA](#)

[ANALYZE TABLE](#) [CHECK TABLE](#) [CHECKSUM TABLE](#) [OPTIMIZE TABLE](#) [REPAIR TABLE](#)

[BACKUP TABLE](#) [RESTORE TABLE](#)

[CACHE INDEX](#) [LOAD INDEX](#) [DESCRIBE](#)

[CREATE | RENAME USER](#) [DROP USER](#) [FLUSH](#) [GRANT](#) [KILL](#) [RESET](#) [REVOKE](#)

Replication Commands

[CHANGE MASTER TO](#) [LOAD DATA FROM MASTER](#) [LOAD TABLE FROM MASTER](#)

[PURGE BINARY LOGS](#) [RESET MASTER](#) [RESET SLAVE](#) [START SLAVE](#) [STOP SLAVE](#)

Data Manipulation Language

SELECT

[Qualifiers](#) [Expression](#) [INTO](#) [FROM and JOIN](#) [WHERE](#)
[ORDER BY](#) [GROUP BY](#) [HAVING](#) [LIMIT](#) [FOR UPDATE](#)

Other DML commands

[DELETE](#) [DO](#) [EXPLAIN](#) [HANDLER](#) [INSERT](#) [LOAD DATA INFILE](#) [LOAD XML](#) [LOCK/UNLOCK](#)
[PREPARE](#) [REPLACE](#) [RLIKE | REGEXP](#) [TRANSACTIONS](#) [TRUNCATE](#) [UNION](#) [UPDATE](#)

Structured Query Language

Structured Query Language (SQL) is a non-procedural computer language, [originally developed](#) in the late 1970s by IBM at its San Jose Research Laboratory.

Let's begin with how to pronounce it. The American National Standards Institute (ANSI) wants it pronounced *ess-kew-ell*. The International Standards Organisation (ISO) takes no position on pronunciation. Many database professionals and most Microsoft SQL Server developers say *see-kwel*. The makers of MySQL prefer *my-ess-kew-ell*. Take your pick.

Although the 'Q' in SQL stands for 'Query', SQL is a language not only for querying data, but for creating and modifying database structures and their contents, for inserting,

updating and deleting database data, for managing database sessions and connections, and for granting and revoking users' rights to all this.

Traditionally, SQL statements specify what a DBMS is to do, not *how* the DBMS is to do it. So SQL is a partial computer language: you cannot use it to produce a complete computer program, only to interface with a database. This you can do in three ways:

- *interactively*: in a standalone application with a MySQL command interface, or
- *statically*: you can embed fixed SQL statements for execution within programs written in other languages (Perl, PHP, Java, etc) or
- *dynamically*: you can [PREPARE](#) SQL statements, and you can use other languages to build runtime SQL statements based on program logic, user choices, business rules, etc., and to send those SQL statements to MySQL.

With many RDBMS products, the line between specifying what the RDBMS is to do, and how it should do it, is blurring; for example MySQL has syntax for telling the query optimiser how to execute a particular query. Since MySQL is an open-source product, you can, in theory, rewrite how MySQL does anything. In practice you are not likely to try that on a large scale. But MySQL has had, traditionally, an interface for writing user-defined functions (UDFs), and with 5.1 MySQL has introduced an advance on this facility—an API for user-coded [plugins](#).

In general a SQL statement ...

- begins with a keyword verb (e.g., SELECT),
- must have a reference to the object of the verb (e.g., * meaning all columns), and
- usually has modifiers (e.g., FROM my_table, WHERE conditional_expression) that scope the verb's action. Modifying clauses may be simple keywords (e.g., DISTINCT), or may be built from expressions (e.g., WHERE myID < 100).

Table 6-1: SQL Statement Components

<i>Component</i>	<i>Type</i>	<i>Use</i>	<i>Examples</i>
verbs	keywords	action descriptors	SELECT, JOIN, UPDATE, COMMIT, GRANT
object, type names	keywords	general object references	TABLE, VIEW, DOMAIN, INTEGER, VARCHAR
function, variable names	keywords	function and variable references	MAX, AVG, SESSION_USER
conjunctions	keywords	conjoin verbs & object references	FROM, WHERE, FOR, THEN
modifiers	keywords	define scope	ANY, TEMPORARY
constant values	keywords	defined constant values	TRUE, FALSE, NULL
identifiers	string literals	names of schemas, databases, tables, views, cursors, procedures, columns, Authorization IDs, etc.	tableName.columnName, "columnName"
operators	symbolic	relate variables and values	<, <=, =, >, >=, LIKE, *
literal values	literals	data	1006, 'Smith', 2005-5-20

Clauses, expressions and statements are built according to a set of simple syntactic rules from keywords (verbs, nouns, conjunctions), identifiers, symbolic operators, literal values

and (except in dynamic SQL) a statement terminator, ';'. Table 6-1 lists the nine kinds of atoms used in SQL to assemble SQL expressions, clauses and statements.

The set of all SQL statements that define schemas and the objects within them, including tables, comprise the *SQL Data Definition Language (DDL)*.

The set of SQL statements that control users' rights to database objects comprise the *Data Control Language (DCL)*. Often DCL is considered part of DDL. SQL statements that store, alter or retrieve table data comprise *Data Manipulation Language (DML)*.

SQL also has:

- *connection statements*, which connect to and disconnect from a database
- *session statements*, which define and manage sessions,
- *diagnostic statements*, which elicit information on the database and its operations,
- *transaction statements*, which define units of work and mark rollback points.

This much, most SQL vendors and users can agree on. But no implementation of SQL is identical to any other. Variation is the exceptionless rule. ANSI and ISO have approved SQL as the official relational query language. ANSI has issued *five* SQL standards: SQL86, SQL89, SQL92, SQL99 and SQL2003. SQL92 remains a common reference point, with three levels: entry level, intermediate and full. SQL99, in contrast, has no levels but rather core and other features: what was *entry-level* in SQL92 becomes *core* in SQL99. Several commercial vendors implement an SQL variant known as Transact-SQL (T-SQL). And so it goes.

The MySQL variant of SQL

MySQL 5 complies with entry-level SQL92; implements some higher-level features of SQL92, T-SQL, SQL99 and SQL2003; and extends ISO SQL in other ways to improve database management, performance and ease of use.

5.0.1 brought views. 5.0.2 brought updateable views and triggers. In later releases of 5.0 and 5.2, these enhancements have become reliable.

MySQL's stated eventual aim is complete ISO SQL compatibility, though with five official definitions of SQL in 17 years, that is [a moving target](#), fully implemented by *no* vendor!¹⁻⁴ Is a SQL feature implemented by no vendor actually SQL? Does SQL consist of the five sets of ISO standards, or the union of all commands implemented by all SQL vendors? Are we to regard a feature implemented by many vendors as "SQL" before it is incorporated into a subsequent version of the standard? Impossible questions, all. Yet we need a usage that makes sense. Here is our rough take. A feature is SQL if it is in one of the five published standards, *or* if it is commonly implemented by vendors. Table 6-2 lists some SQL-compliant functionality added to MySQL 5. 6.0 brings Falcon, a new ACID-compliant storage engine. Still,

Table 6-2: More SQL compliance

Command	Release
INFORMATION_SCHEMA	5.0.2
Stored routines	5.0.0
Triggers	5.0.2
Views	5.0.1
XA transactions (InnoDB)	5.0.3
Generalised tablespaces	5.1

basic features like CHECK CONSTRAINT, CREATE ASSERTION, per-statement triggers, and schemas within databases are missing from the roadmap.

To read the rest of this and other chapters, [buy a copy of the book](#)

[TOC](#) [Previous](#) [Next](#)
