Writing portable SQL

How to write queries that run on every database Exponential supports - escaping and casting, limits as parameters, the dialect differences of each driver (row limits, ids, concatenation, MD5, bitwise operators, joined...

Source file: `doc/eng-us/help/software-engineers/writing-portable-sql.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/software-engineers/writing-portable-sql.md

This page is for engineers who write SQL in kernel or extension code. Exponential runs on six databases, so a query that works on your development database is not finished until it works on the others. The rules and the dialect table come from Database and SQL standards.

Contents

1. The two mandatory rules
2. Prefer the API to SQL
3. Dialect differences
4. Patterns that stay portable
5. Testing on more than one engine

The two mandatory rules

1. **Escape every string** with `$db->escapeString( $value )`, inside single quotes. Each driver uses the engine's own function; the Oracle driver doubles single quotes.
2. **Cast every number** with `(int)` before it goes into SQL. A cast cannot be escaped around.

These rules are what the 6.0.13 security fixes restored in the kernel (integer casts, `escapeString()`, a column whitelist for `ORDER BY`); the same patterns in extension code are not fixed by a kernel update (Security hardening of 6.0.13).

Prefer the API to SQL

- `eZPersistentObject::fetchObjectList()` and the other persistent-object methods escape for you.
- Content queries go through fetch functions or the content classes; they apply permissions and work on every engine, MongoDB included.
- Pass offset and limit as the parameters of `arrayQuery()`, so each driver writes its own syntax.

Dialect differences

Topic	SQLite 3	PostgreSQL	MySQL, MariaDB	Oracle	MongoDB adapter

Topic	SQLite 3	PostgreSQL	MySQL, MariaDB	Oracle	MongoDB adapter
Row limit	`LIMIT offset, count`	`LIMIT count OFFSET offset`	`LIMIT offset, count`	`OFFSET n ROWS FETCH NEXT m ROWS ONLY`	`limit` and `skip`
New ids	`lastInsertRowID()`	sequence `<table>_<column>_seq`	`AUTO_INCREMENT`	sequence `s_<table>`	a counter document per sequence
String concatenation	`||`	`||`	`concat( ... )`	`||`	`concat( ... )`, read by the adapter
MD5	user function `md5()`	needs `pgcrypto`	`MD5()`	`STANDARD_HASH( s, 'MD5' )`	hashed in PHP
MySQL `UPDATE ... JOIN`	rewritten (needs SQLite 3.33)	not rewritten	native	not rewritten	not applicable
Empty string	`''`	`''`	`''`	stored as NULL	not checked
String comparison	= with case	with case	without case (default collations)	with case	with case unless a collation is set
Start of a transaction	`BEGIN IMMEDIATE` after a writer lock	`BEGIN WORK`	`BEGIN WORK`	no auto commit	none

The full table, with sources, is in Dialect differences, driver by driver.

Patterns that stay portable

Instead of	Write
`LIMIT 10` in the SQL string	`array( 'offset' => 0, 'limit' => 10 )` as the parameter
`UPDATE a JOIN b ...`	`UPDATE a SET ... WHERE id IN ( SELECT ... )`, or two statements in a transaction
Comparing to `''`	handle `''` and `NULL` alike, because Oracle stores the empty string as `NULL`
Relying on case-insensitive =	normalize case in PHP or on both sides
`SET @var := ...`	read the value in PHP and pass it into the next statement
An engine function (`GROUP_CONCAT`, `ILIKE`)	do the work in PHP, or branch on the driver with a fallback for every engine
`ORDER BY` a column name from the request	a whitelist of allowed columns

Testing on more than one engine

- SQLite needs no server: run your code against an SQLite installation as a second engine.
- CI runs the database tests on MySQL 8.0 and PostgreSQL 15 (PHP 8.3); reproduce them with `tests/runtests.php --dsn ... --db-per-test` against a throwaway database (CI gates).
- Tests that need a live database read its location from environment variables and skip when it is not set (DB-less tests and test suites).

Related pages

- Persistence and the database layer
- Database code for PHP 8
- Chapter 9, cross-engine notes
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.