SQL with eZDB, safely

Writing SQL in Exponential with eZDB - instance(), query() and arrayQuery() with limit and offset, escapeString() for strings and (int) casts for numbers, generateSQLINStatement(), transactions with begin(), commit()…

Source file: `doc/eng-us/help/php-developers/sql-with-ezdb.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/sql-with-ezdb.md

This page is for PHP developers who write SQL. Exponential reaches every database through `eZDB` (`lib/ezdb/classes/ezdb.php`), which returns a driver implementing `eZDBInterface` (`lib/ezdb/classes/ezdbinterface.php`). The driver interface has no bound parameters for arbitrary SQL, so safety comes from two habits: escape every string, cast every number.

Contents

1. Getting the connection
2. Reading
3. Writing
4. Escaping: the two rules
5. Lists of values
6. Transactions
7. Return values
8. When not to write SQL

Getting the connection

PHP

```php
$db = eZDB::instance();
```

`eZDB::instance()` returns the shared connection for the current siteaccess's `[DatabaseSettings]`. Do not create your own connection with `forceNewInstance` unless you really need a second one; it costs a connection and escapes the transaction state of the request.

Reading

PHP

```php
$userID = (int)$userID;
$rows = $db->arrayQuery(
    "SELECT id, title, created FROM acme_report_item WHERE user_id = $userID ORDER BY created DES
C",
    array( 'offset' => 0, 'limit' => 20 )
);
foreach ( $rows as $row )
{
    // $row['id'], $row['title'], $row['created'] are strings (or null)
}

$titles = $db->arrayQuery( "SELECT title FROM acme_report_item WHERE user_id = $userID",
                           array( 'column' => 'title' ) );     // a flat list of one column
```

Pass `limit` and `offset` as parameters instead of writing `LIMIT` into the SQL: the driver writes the syntax its database understands (Oracle and SQLite differ from MySQL).

Writing

PHP

```php
$title = $db->escapeString( $title );
$now = time();
$db->query( "INSERT INTO acme_report_item ( user_id, title, created ) VALUES ( $userID, '$title',
 $now )" );
$newID = $db->lastSerialID( 'acme_report_item', 'id' );
```

Escaping: the two rules

Value	Rule	Example
string	`$db->escapeString()`, inside single quotes	`"... WHERE title = '" . $db->escapeString( $title ) . "'"`
integer	`(int)`	`"... WHERE id = " . (int)$id`
float	`(float)`	`"... WHERE amount > " . (float)$min`
column or table name from input	an allow list, never the input itself	`in_array( $sort, array( 'title', 'created' ), true )`
sort direction	a fixed choice	`$dir = $desc ? 'DESC' : 'ASC';`

Cast or escape as close to the SQL as possible, even when the value "was already checked": the next change to the code around it may remove the check. Values from `$Params`, `eZHTTPTool`, settings, imports and the database itself are all input (Validating input and casting).

Lists of values

PHP

```php
$ids = array_map( 'intval', $ids );
$in = $db->generateSQLINStatement( $ids, 'id', false, true, 'int' );   // id IN ( 1, 2, 3 ); driv
ers may override it
$rows = $db->arrayQuery( "SELECT id, title FROM acme_report_item WHERE $in" );
```

With a type (`'int'`), `generateSQLINStatement()` casts every element through `implodeWithTypeCast()`; let the driver build the statement rather than concatenating it yourself, since a driver can write it differently for its database.

Transactions

PHP

```php
$db->begin();
$item->store();
$log->store();
$db->commit();
```

- `begin()` and `commit()` nest: an inner pair inside an outer one only counts (`transactionCounter()`), and the outer `commit()` commits. Every `begin()` needs exactly one `commit()` or `rollback()` on every path, including early returns and exceptions.
- Transactions are used only when `[DatabaseSettings] Transactions=enabled`.
- Keep transactions short; on SQLite a writer waits for the writers ahead of it (SQLite transactions).

Return values

Call	Returns
`query()`	`true` or `false` for most statements
`arrayQuery()`	an array of rows (empty when nothing matches)
`lastSerialID()`	the new id after an insert

Check `query()`'s result when it matters. Do not rely on `SET @var := ...` session variables through `query()`; for inserts that need the new id, `eZPersistentObject::store()` sets it for you. Results also differ across engines in edge cases, for example an empty result and a failed statement on the MongoDB adapter (Code for every supported database).

When not to write SQL

Need	Better
rows of your own table	`eZPersistentObject` conditions, escaped for you (Persistent object definitions in depth)
content	`eZContentObjectTreeNode::subTreeByNodeID()` and the content API (The content API in PHP)
the schema	`.dba` files and the schema handler (Database schema and .dba files)

Related pages

- eZDB reference
- Database access and SQL
- Databases and SQL standard

- 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.