Database support

Which databases Exponential 6.0.x supports and in which order (SQLite first, then PostgreSQL, MongoDB, MariaDB, Oracle and MySQL), with the versions, PHP extensions, settings, installers, character sets, performance…

Source file: `doc/eng-us/help/database-support.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/database-support.md

This page is for administrators who choose, install and run the database of an Exponential site, and for developers who write code that has to work on every database. It answers four questions in one place: which databases Exponential 6.0.x supports, in which order they are recommended, what each one needs, and how well each one is tested. Every fact on this page was checked against the code and the continuous integration of the 6.0.15 line; where something could not be verified, the page says so.

The step-by-step operator's chapter, with every setting explained, is chapter 9 of the installation guide, Databases. This page is the overview and the support statement; it links to that chapter wherever the details are.

Contents

1. The supported databases, in order
2. At a glance
3. Versions, checked against the code
4. PHP extensions
5. The settings: DatabaseSettings in site.ini
6. Installer support per path
7. SQLite 3, the default
8. PostgreSQL
9. MongoDB
10. MariaDB
11. Oracle
12. MySQL
13. Microsoft SQL Server: possible with additional work
    - A starting point: the xrowmssql extension
14. Character sets and collations
15. Performance and scaling
16. Backup and restore
17. Moving a site to another database
18. Test coverage (CI)
19. Which database is my site using?

20. Related pages

The supported databases, in order

> **Supported databases (Exponential 6.0.x), in order of preference:** 1. SQLite 3 (the default) · 2. PostgreSQL · 3. MongoDB · 4. MariaDB · 5. Oracle · 6. MySQL. Microsoft SQL Server is known to be possible with additional work at this time.

This is the project's official order for Exponential 6.0.x. It is the order in which the databases are recommended, documented and listed throughout the documentation. All six are supported: every one of them has a driver in the kernel or in a shipped extension, an entry in the setup wizard, and a path in the command-line installers.

What the order means in practice:

1. **SQLite 3** is the default. The setup wizard preselects it, `exp:install` uses it when no `--db` is given, and the kickstarter's template names it first. It needs no database server: the whole content repository is one file. With the driver Exponential ships it is a full production database for a site served from one machine.
2. **PostgreSQL** is the recommended database server: strict SQL, row-level locking, replication and point-in-time recovery of its own, and it is tested in CI on every push.
3. **MongoDB** is supported through an adapter that translates the kernel's SQL-shaped calls into MongoDB operations. It is the choice for organizations standardized on MongoDB; read its limits before you choose it.
4. **MariaDB** runs on the same driver as MySQL (`mysqli`). It is the MySQL-compatible server the project prefers.
5. **Oracle** (19c and later) is supported through the `ezoracle` extension, which `composer.json` requires, so it is present in every installation.
6. **MySQL** is supported on the same driver as MariaDB and is tested in CI. It is last in the order, not because it works less well, but because the project recommends the others first for new sites.

Microsoft SQL Server has no driver in Exponential 6.0.x. It is known to be possible: the database layer is a small, documented interface, and everything that would have to be written is listed in Microsoft SQL Server. An extension written for the 4.x line, `se7enxweb/xrowmssql`, is the natural starting point for that work.

At a glance

	1. SQLite 3	2. PostgreSQL	3. MongoDB	4. MariaDB	5. Oracle	6. MySQL
DatabaseImplementation	sqlite3	ezpostgresql	mongodb	ezmysqli	ezoracle	ezmysqli
Driver class	eZSQLite3DB	eZPostgreSQLDB	expMongoDB	eZMySQLiDB	eZOracleDB	eZMySQLiDB

	1. SQLite 3	2. PostgreSQL	3. MongoDB	4. MariaDB	5. Oracle	6. MySQL
PHP extension	`sqlite3`	`pgsql`	`mongodb` + library `mongodb/mongodb`	`mysqli`	`oci8`	`mysqli`
Database server	none (one file)	yes	yes	yes	yes	yes
Installer default	yes					
Setup wizard	yes	yes	yes (slow step)	yes	yes	yes
`exp:install --db=`	`sqlite` (default)	`pgsql`	`mongodb`	`mariadb`	`oracle`	`mysql`
Several web servers on one database	no	yes	yes	yes	yes	yes
Character set	UTF-8	`UTF8`	UTF-8 (BSON)	`utf8mb4`	`AL32UTF8`	`utf8mb4`
Schema update files	`update/database/sqlite/`	`update/database/postgresql/`	none (no SQL)	`update/database/mysql/`	in `ezoracle`	`update/database/mysql/`
Tested in the project's CI	driver unit tests	yes (PostgreSQL 15)	adapter unit tests	no	no	yes (MySQL 8.0)

Versions, checked against the code

Three different numbers exist for each database, and they are easy to mix up:

- **Wizard floor**: the oldest server version the setup wizard accepts, from `eZSetupDatabaseMap()` in `kernel/setup/ezsetupcommon.php`. It is a technical floor kept from earlier releases, not a recommendation.
- **Practical minimum**: the oldest version the current drivers really need, as stated by the driver code and the driver documentation.
- **Seen working**: the versions the project runs on its development server and in CI.

Database	Wizard floor	Practical minimum	Seen working
SQLite 3	3.0.1	**3.33** (the driver rewrites MySQL's `UPDATE ... JOIN` into `UPDATE ... FROM`, which SQLite has from 3.33; `eZSQLite3DB::rewriteJoinedUpdate()`)	library 3.34.1 in PHP 8.5 on the development server; the SQLite library of the CI images (version not pinned)
PostgreSQL	8.0	**13** (the installer creates `pgcrypto`, a trusted extension from PostgreSQL 13 on); from 15 on the site's user must own the database	13.23 on the development server; 15 in CI
MongoDB	4.0	**6.0** server, PHP **8.2** or later, PHP `mongodb` extension **1.18** or later (MongoDB as the database)	server 8.3.13 with extension 2.3.3 on PHP 8.5 on the development server

Database	Wizard floor	Practical minimum	Seen working
MariaDB	4.1.1 (the MySQL number)	a release your vendor still supports; `utf8mb4` and InnoDB	10.5.29 on the development server
Oracle	19.0	**19c** (the driver uses `STANDARD_HASH` and `OFFSET`/`FETCH`; 11g and older are not supported)	Oracle AI Database 26ai Free (23.26) with oci8 3.4.1 and Instant Client 23.26 on PHP 8.5
MySQL	4.1.1	a release your vendor still supports; `utf8mb4` and InnoDB	8.0 in CI

Use a server version your vendor still supports with security fixes. The floors exist so that the wizard does not refuse an old server outright; they are not a promise that a server of that age works with the current drivers.

PHP. Exponential 6.0.x runs on PHP 8.0 to 8.5 (PHP 7.4 usable without support) (`composer.json`, `"php": "^7.4 || ... || ^8.8"`). The database drivers follow that range, with one exception: the MongoDB adapter needs PHP 8.2 or later. Velocity, the recommended web server, needs PHP 8.1. See chapter 2.

PHP extensions

One database extension is required; the setup wizard's critical test `database_extensions` passes when at least one of `sqlite3`, `mysqli`, `pgsql`, `mongodb` and `oci8` is loaded (`settings/setup.ini`, `[database_extensions] Extensions=sqlite3;mysqli;pgsql;mongodb;oci8`, `Require=one`). Install only the one your site uses.

Database	Extension	Where it comes from
SQLite 3	`sqlite3`	bundled with PHP; RHEL family `php-pdo` (it also carries `pdo_sqlite`), Debian and Ubuntu `php<version>-sqlite3`. `pdo_sqlite` is not needed by the kernel; the Exponential Platform installers use it.
PostgreSQL	`pgsql`	bundled with PHP; `php-pgsql`, `php<version>-pgsql`
MongoDB	`mongodb` (PECL) and the Composer library `mongodb/mongodb`	`pecl install mongodb` or `php-pecl-mongodb`; the library with `composer require mongodb/mongodb` (`composer.json` only suggests it). CI installs `mongodb/mongodb ^1.21 || ^2.0`.
MariaDB, MySQL	`mysqli`	bundled with PHP (mysqlnd); `php-mysqlnd`, `php<version>-mysql`
Oracle	`oci8` (PECL)	PECL oci8 3.4 for PHP 8.4 and 8.5, 3.2 for PHP 8.1 to 8.3, built against an Oracle client (the Instant Client is enough); EPEL, Remi or your vendor

Check what a PHP has with `php -m`, and remember that the web server's PHP and the command line PHP are configured separately: check both. The optional wizard test `database_all_extensions` lists the database extensions that are missing; it is information only.

The settings: DatabaseSettings in site.ini

The connection is configured in `settings/site.ini`, block `[DatabaseSettings]`, and overridden (never edited in place) in `settings/override/site.ini.append.php` or a siteaccess's `site.ini.append.php`. The installers write the override for you. `DatabaseImplementation` names the driver by an alias that `ImplementationAlias[]` maps to a class:

Alias in `DatabaseImplementation`	Class
`sqlite3`	`eZSQLite3DB` (`lib/ezdb/classes/ezsqlite3db.php`)
`ezpostgresql`, `postgresql`, `pgsql`	`eZPostgreSQLDB` (`lib/ezdb/classes/ezpostgresqldb.php`)
`mongodb`	`expMongoDB` (`lib/ezdb/classes/expmongodb.php`)
`ezmysqli`, `mysqli`, `ezmysql`, `mysql`	`eZMySQLiDB` (`lib/ezdb/classes/ezmysqlidb.php`)
`ezoracle`, `oracle`	`eZOracleDB` (in the `ezoracle` extension, registered by its own `site.ini.append.php`)

The examples below use placeholder names and passwords; put your own values in your override file and never commit that file. Every key is explained in 9.1 How Exponential chooses a database driver.

1. SQLite 3

INI

```ini
[DatabaseSettings]
DatabaseImplementation=sqlite3
# a plain name lives in var/storage/sqlite3/; a name starting with / is an absolute path
Database=exponential.db
# seconds a transaction waits for the writers ahead of it; keep it below the web server's request
 timeout
SQLiteTransactionWait=25
# PRAGMAs over the driver's defaults, one name=value per line (empty keeps the defaults)
SQLitePragmas[]
```

2. PostgreSQL

INI

```ini
[DatabaseSettings]
DatabaseImplementation=ezpostgresql
Server=localhost
Port=5432
User=exponential
Password=<the password>
Database=exponential
UsePersistentConnection=disabled
```

3. MongoDB

INI

```ini
[DatabaseSettings]
DatabaseImplementation=mongodb
Server=localhost
Port=27017
# required, also on a server without access control
User=exponential
Password=<the password>
Database=exponential
```

4. MariaDB and 6. MySQL

INI

```ini
[DatabaseSettings]
DatabaseImplementation=ezmysqli
Server=localhost
# empty: 3306
Port=
User=exponential
Password=<the password>
Database=exponential
# a Unix socket instead of host and port
Socket=disabled
Charset=utf-8
UsePersistentConnection=disabled
```

5. Oracle

INI

```ini
[DatabaseSettings]
DatabaseImplementation=ezoracle
User=exponential
Password=<the password>
# the connect string: Easy Connect host:port/service_name, a TNS alias or a full descriptor;
# Server and Port are not used by this driver
Database=127.0.0.1:1521/FREEPDB1
Charset=utf-8

[ExtensionSettings]
ActiveExtensions[]=ezoracle
```

After a change, clear the INI cache so every process reads it, and restart Velocity's workers if Velocity serves the site:

Shell

```shell
php bin/php/ezcache.php --clear-tag=ini --allow-root-user
php bin/php/console exp:velocity restart --allow-root-user
```

Read the effective value with `php bin/php/console exp:ini get site.ini/DatabaseSettings/DatabaseImplementation`. A mistyped alias does not stop the request at once: the kernel falls back to a null database and logs "Database implementation not supported" in `var/log/error.log`, and every content page then fails. Read `error.log` first when a site behaves as if its database were down.

Installer support per path

Exponential has three installers in the kernel. All three support all six databases.

Database	Setup wizard (browser)	Kickstarter (`kickstart.ini`, `[database_choice] Type=`)	`exp:install --db=`
1. SQLite 3	yes, listed first and preselected (`settings/setup.ini [DatabaseSettings] DefaultType=sqlite3`); proposes `sqlite.db`	`sqlite3` (the default in `kickstart.ini-dist`)	`sqlite` (the default; also `sqlite3`), file `exponential.db`
2. PostgreSQL	yes; prefills port 5432 and user `postgres`; creates `pgcrypto` or tells you how	`pgsql` (`postgresql` accepted)	`pgsql` (also `postgresql`, `postgres`)
3. MongoDB	yes; the configuration step can take one to three minutes, so raise PHP-FPM's `request_terminate_timeout` or use the console	`mongodb`	`mongodb` (also `mongo`)
4. MariaDB	yes, as "MySQL Improved"; checks that the database is UTF-8	`mysqli` (`mysql` accepted)	`mariadb` (the same as `mysql`)
5. Oracle	yes, when `oci8` is loaded and `extension/ezoracle` is present; activates the extension for the new site	`oci8` (`oracle` accepted); `Database` is the connect string	`oracle` (also `oci8`, `ezoracle`); `--db-name` is the service name, default `FREEPDB1`
6. MySQL	yes, as "MySQL Improved"	`mysqli` (`mysql` accepted)	`mysql` (also `mysqli`)

- If the `sqlite3` extension is missing, the wizard chooses the first database whose extension is loaded and says so.
- None of the installers creates a database server's database or user. Create them first; the per-database sections below show how.
- `exp:install` takes the password from the environment variable `EXP_INSTALL_DB_PASSWORD`, so it stays out of the shell history and the process list.
- **Demo installs.** This repository has no separate demo installer; demonstration sites are installed with the same commands (`exp:install` or the kickstarter), usually on SQLite.
- **Exponential Platform** (the Symfony-based sister product, not the 6.0.x kernel) has its own SQLite installer that uses `pdo_sqlite`; see SQLite for Exponential Platform.

The installers are described in chapter 5, the setup wizard, chapter 6, the kickstarter and chapter 7, the console install.

1. SQLite 3, the default

What it is. The whole content repository in one file below `var/storage/sqlite3/`, opened by the driver `eZSQLite3DB`. No server process, no network, no user accounts to manage.

What the driver adds to SQLite. Write-ahead logging (WAL), so page views keep reading while an editor publishes; connection PRAGMAs chosen for a web server (`synchronous=NORMAL`, a 64 MB page cache, a 256 MB memory map, a 5-second busy timeout); the functions SQLite lacks (`md5()`, the `BIT_OR` and `BIT_AND` aggregates the content engine uses for language

masks); a rewrite of MySQL's `UPDATE ... JOIN`; and, since 6.0.15, a **write queue**: every transaction takes an exclusive lock on `<database>.writer-lock` and then runs `BEGIN IMMEDIATE`, so concurrent publishes wait their turn (up to `SQLiteTransactionWait` seconds, default 25) instead of failing with "database is locked". MySQL-only session statements (`SET NAMES`, `SET FOREIGN_KEY_CHECKS` and similar) are accepted and ignored.

When it fits. One machine, any number of readers, an editorial team. The project's own load tests had concurrent readers scale with the number of processes and 16 concurrent writers complete without errors, queued. Choose a database server instead when more than one machine must write the database (WAL needs shared memory on one host, so a network file system is not a substitute), when many editors publish large changes at the same moment all day, or when you need replication or database-level users and grants.

Files and permissions. The database is four files: the database, `-wal`, `-shm` and `.writer-lock`. Every process that uses the site (the web server's PHP, Velocity's workers, cronjobs, console commands) must be able to write the directory and the files; give them one common group, the directory mode `2775` and a umask of `0002`. Never serve the directory over HTTP (the shipped rewrite rules do not).

Limits. One host; one writer at a time (queued); no replication of its own; access control is the file system's.

The full treatment, with every PRAGMA, the queue's behavior under load, monitoring and maintenance (`ANALYZE`, `VACUUM`, the WAL file), is 9.2 SQLite 3: a production database in one file. The task-oriented introduction is SQLite database support, the driver specification The SQLite3 database driver.

2. PostgreSQL

What it is. The driver `eZPostgreSQLDB` on the PHP extension `pgsql`.

Create the database. Make the site's user the **owner** of the database (from PostgreSQL 15 on, other users may no longer create tables in its `public` schema):

Shell

```
sudo -u postgres createuser --pwprompt exponential
sudo -u postgres createdb --owner=exponential --encoding=UTF8 --template=template0 exponential
```

pgcrypto. The driver computes MD5 with `digest()` from `pgcrypto`. The installer runs `CREATE EXTENSION IF NOT EXISTS pgcrypto`; as a trusted extension (PostgreSQL 13 and later) the database owner may create it. If the server lacks it, install your distribution's PostgreSQL contrib package and create it as the owner or a superuser. The wizard's message when it is missing reads "The 'digest' function is not available in your database, and Exponential cannot run without it."

Sequences. Auto-increment columns use sequences named `<table>_<column>_seq`. After loading data with explicit ids, correct them (`SELECT setval('ezcontentobject_id_seq', (SELECT max(id) FROM ezcontentobject));`); the schema installer does this itself after it loads data.

Limits. None specific to Exponential's own code. Extension code written for MySQL (backtick quoting, MySQL-only `SET` statements) fails here; see cross-engine notes.

Details: 9.4 PostgreSQL.

3. MongoDB

What it is. The adapter `expMongoDB` on the PHP extension `mongodb` and the library's `MongoDB\Client`. It translates the kernel's SQL-shaped calls into `find`, `aggregate`, insert, upsert and delete operations. MongoDB arrived with 6.0.14; its schema handler is `expMongoSchema` (`settings/dbschema.ini`, type `mongo`).

Create the user in the site's own database (the adapter always puts `<user>:<password>@` into the connection URI, and MongoDB uses the database in the URI as the authentication database):

Shell

```
mongosh --eval '
  db.getSiblingDB("exponential").createUser({
    user: "exponential",
    pwd: passwordPrompt(),
    roles: [ { role: "readWrite", db: "exponential" }, { role: "dbAdmin", db: "exponential" } ]
  })'
```

After installing, create the indexes the kernel's queries rely on (`mongosh "mongodb://exponential@localhost:27017/exponential" --file bin/mongodb/create_indexes.js`), then run the cronjobs once to build the search index.

MongoDB limits

From MongoDB as the database and section 26 of the porting reference:

- The adapter never runs real SQL. Hand-written SQL that it cannot translate returns an empty result and logs a warning (search the logs for `MONGO TODO`). Extensions with SQL of their own may need a MongoDB path.
- Deep URL alias regeneration after moving subtrees is incomplete; role and policy list pages can be empty although permissions are enforced; some cronjobs are not ported; `eztags` is not usable; the shop checkout and `ezflow` scheduling are untested end to end.
- The SQL query cache does not apply.
- There are no SQL update files. The upgrade check compares collections instead of tables and prints the `mongosh` commands that create the missing ones.

Details: 9.5 MongoDB.

4. MariaDB

What it is. The same driver as MySQL, `eZMySQLiDB`, on the PHP extension `mysqli`. Everything in 6. MySQL applies; the notes here are what differs.

- Install with `exp:install --db=mariadb` (the same as `--db=mysql`) or choose "MySQL Improved" in the wizard.
- Create the database in `utf8mb4` with an InnoDB default, as for MySQL. In MariaDB, as in MySQL, the connection name `utf8` means the three-byte `utf8mb3`; see Character sets and collations.
- Backups with `mariadb-dump` (or the older name `mysqldump`) `--single-transaction`.
- MariaDB is not in the project's CI; the MySQL 8.0 job exercises the same driver. MariaDB 10.5 runs on the development server.

5. Oracle

What it is. The `ezoracle` extension (version 2.3.7, required by `composer.json` as `~2.3.7`) provides the driver `eZOracleDB`, the schema handler `eZOracleSchema`, console commands (`ext:ezoracle:<name>`) and cronjobs. Because `composer.json` requires it, it is in `extension/ezoracle` in every installation; it is activated only for sites that use Oracle.

Requirements. Oracle Database 19c or later in the character set `AL32UTF8`; PHP with `oci8` built against an Oracle client (the Instant Client is enough; `ORACLE_HOME` is not needed with it); a user that may create sessions, tables, views, triggers, sequences and procedures. `php extension/ezoracle/bin/php/ora-grant.php` prints the grants for a DBA to run.

Connections. `ezoracle.ini [ConnectionSettings]` adds persistent connections, Database Resident Connection Pooling (DRCP, so hundreds of PHP workers share a pool of server processes), retries, failover, call timeouts and session tracing. Every feature is off by default. When the settings come from a site extension instead of `settings/override/`, list that extension **before** `ezoracle` in `ActiveExtensions[]`.

Oracle's own behavior to know. Oracle stores `''` as NULL (`OracleEmptyStringForNull=enabled` returns `''` for text columns); it compares strings case-sensitively (`OracleCaseInsensitive=enabled` with linguistic indexes from `ext:ezoracle:ci-indexes --create`); and it rejects `AS` before a table alias.

Limits. Exponential 6.0.15 or later is needed (earlier versions do not offer Oracle in the installers). Oracle is not in the project's CI; Oracle-only tests in the kernel suite skip on other engines. The SQL query cache applies.

Details: 9.6 Oracle 19c and later, and the extension's `README.md` and `INSTALL` in `extension/ezoracle/`.

6. MySQL

What it is. The driver `eZMySQLiDB` on `mysqli`, the engine most third-party extensions were originally written against.

Create the database and the user. No global privilege is needed:

SQL

```
CREATE DATABASE exponential CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'exponential'@'localhost' IDENTIFIED BY '<a strong password>';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER,
      LOCK TABLES, CREATE TEMPORARY TABLES
   ON exponential.* TO 'exponential'@'localhost';
```

InnoDB. The installer creates InnoDB tables when the server offers them; transactions need InnoDB. `php bin/php/ezconvertmysqltabletype.php --list` lists the table types of an older installation, `--newtype=InnoDB` converts them.

Read replicas. `UseSlaveServer=enabled` with `SlaveServerArray[]` and the related keys sends reads to a replica chosen at random per request; writes go to `Server`. This is the only engine with built-in read splitting.

Limits. The connection character set `utf-8` maps to MySQL's `utf8` (three bytes), so characters outside the Basic Multilingual Plane, such as emoji, cannot pass through the connection even when the tables are `utf8mb4`.

Details: 9.3 MySQL and MariaDB.

Microsoft SQL Server: possible with additional work

The statement. Microsoft SQL Server is known to be possible with additional work at this time. It is not supported in Exponential 6.0.x: there is no driver, no schema handler and no installer entry for it, and nothing on this page should be read as a promise of a release that has one.

What exists today.

- **A small driver interface.** Every database is reached through `eZDBInterface` (`lib/ezdb/classes/ezdbinterface.php`), and the five existing drivers show how to implement it for engines as different as a file database, three SQL servers and a document store. The MongoDB adapter in particular proves that the kernel does not depend on one SQL dialect.
- **An engine-neutral schema.** The kernel's schema and initial data are kept in the generic `.dba` format (`share/db_schema.dba`, `share/db_data.dba`), which a schema handler turns into the engine's own DDL.
- **Pluggable schema handlers.** `settings/dbschema.ini` maps a type to a handler class (`SchemaPaths[]`, `SchemaHandlerClasses[]`), so a new engine is registered by settings, not by changing the kernel.
- **Pluggable driver aliases.** `site.ini [DatabaseSettings] ImplementationAlias[]` is extended from an extension's `site.ini.append.php`, the way `ezoracle` adds `ezoracle` and `oracle`.
- **Pluggable installer types.** The kickstarter template says "other values are possible if provided by extension".
- **Tools that already read SQL Server.** The database manager of the `sevenx_dse` extension (AdminNeo, a fork of Adminer) ships an MS SQL driver for browsing and editing tables, and the generic SQLAlchemy exporter in section 18.6 of the MongoDB porting reference can read SQL Server through `pyodbc`. Neither is a way to run a site on SQL Server.
- **PHP drivers.** Microsoft publishes the `sqlsrv` and `pdo_sqlsrv` extensions for PHP on Linux and Windows. Check Microsoft's support matrix for the PHP versions a release covers; this has not been tested with Exponential.

A starting point: the xrowmssql extension

A SQL Server extension already exists, written for earlier editions of the database API: se7enxweb/xrowmssql, "A full-featured open source Microsoft SQL Server database plugin" (repository description). Its README calls it the "Microsoft SQL Server database extension for eZ Publish 4.x and SQL Server 2005/2008", "Developed by xrow GmbH", and says it includes a database driver and a database schema definition. The repository holds the code as copied from `xrow/xrowmssql` in October 2013; since then only funding files were added (its `git log`). It has not been run on Exponential 6, and this page does not claim that it works there.

What it provided (read from its `classes/`, `settings/`, `sql/`, `patch/` and `docs/` on 18 October 2026):

Part	File	What it did

Part	File	What it did
Driver	`classes/ezmssqldb.php` (`eZMSSQLDB`)	An `eZDBInterface` driver: connect, query, `arrayQuery()`, transactions, `lastSerialID()` through `SELECT @@identity`, `escapeString()` (doubled quotes), `subString()` (`substring()`), `concatString()` (the `+` operator), `md5()` (`HashBytes('md5', ...)` converted to hex), `eZTableList()` from `sysobjects`, database lists and server version
Connection layer	`classes/ezmssqlconnector.php` (`eZMSSQLFunctions`)	Two back ends: Microsoft's `sqlsrv` extension on Windows, and the old `mssql_*` functions over FreeTDS (`dblib`) on Linux
Unicode literals	`eZMSSQLDB::_appendN()`	Prefixed non-numeric string literals with `N`, so text went into `NVARCHAR` columns as Unicode
Identity inserts	`eZMSSQLDB::query()`	Switched `IDENTITY_INSERT` on and off around inserts that set the key column
Schema handler	`classes/ezmssqlschema.php` (`eZMssqlSchema`)	Turned `.dba` into SQL Server DDL (`nvarchar`, `ntext`, `int IDENTITY(0,1)`, primary keys and indexes, insert triggers) and read an existing database back into a schema
Settings	`settings/site.ini.append`, `settings/dbschema.ini.append`	The aliases `mssql` and `ezmssql` for `ImplementationAlias[]`, port 1433, the schema handler under `SchemaHandlerClasses[mssql]`, and per-column type and `NOT NULL` translations for long text columns
Server objects	`sql/PROCEDURE.sql`, `sql/FUNCTION.sql`, `sql/clustering.sql`	A stored procedure and function to create in SQL Server first, and the cluster tables
Kernel patch	`patch/4.3/kernel.patch`	A change to `kernel/classes/ezcontentlanguage.php` for SQL Server's arithmetic on language masks
Install notes	`README`, `docs/`	Client setup (SQL Server Native Client on Windows, FreeTDS on Linux), loading the schema with `ezsqlinsertschema.php --type=mssql`, troubleshooting and known bugs

Its `extension.xml` names version 1.0.0 and "GNU General Public License v2.0", while the repository's `LICENSE` file is the GPL version 3; check both before reusing code. The settings expect the extension directory to be named `extension/mssql`.

How it maps to the work list below, and what would need modernizing:

Work list row	What xrowmssql already has	What would need to change
A driver class	Most methods of `eZDBInterface`	PHP 8: its PHP 4 style constructor (`function eZMSSQLDB()` calling `$this->eZDBInterface()`) does not run on PHP 8, where `eZDBInterface` has only `__construct()`; `connect()` lacks the `$port` parameter of today's drivers; the `mssql_*` functions were removed from PHP 7 on, so only `sqlsrv` remains as a back end, on Linux too
Offset and limit	Nothing in use: the `ROW_NUMBER()` rewrite in `arrayQuery()` is commented out, so `limit` and `offset` are ignored	`ORDER BY ... OFFSET n ROWS FETCH NEXT m ROWS ONLY`, which needs SQL Server 2012 or later
`lastSerialID()`	`@@identity`	`SCOPE_IDENTITY()`, which is not changed by triggers, or `OUTPUT inserted.id`

Work list row	What xrowmssql already has	What would need to change
`md5()`, `subString()`, `concatString()`	Present	`CONVERT(VARCHAR(32), HASHBYTES('MD5', ...), 2)` instead of `master.dbo.fn_varbintohexstr`; `NVARCHAR(MAX)` instead of the 8000 character cap
`bitAnd()`/`bitOr()`	The kernel's defaults (the bitwise AND and OR operators) already fit SQL Server	Nothing
The `BIT_OR` aggregate	Not present	Still to do, as described below
`UPDATE ... JOIN`	Not present	Still to do (`UPDATE t SET ... FROM t JOIN ...`)
A schema handler	`eZMssqlSchema`, both directions	PHP 8 constructor; `NVARCHAR(MAX)` instead of the deprecated `ntext`; a review of the column translations against today's `share/db_schema.dba`
SQL files, installer entries, extensions, tests and CI	Install notes only	All still to do
Session statements, case and collation	Not handled	Still to do

Taken together, xrowmssql shows how each piece was solved once for SQL Server and gives a schema handler and a driver skeleton to start from; making it run on Exponential 6 means porting it to PHP 8 and `sqlsrv`, replacing its SQL Server 2005 idioms, and adding the rows it never covered.

What is missing, the work list.

Piece	Where	What it involves
A driver class	`lib/ezdb/classes/` or an extension (as `ezoracle` does it)	implement `eZDBInterface` on `sqlsrv`: connect, query, `arrayQuery()` with offset and limit (`OFFSET ... FETCH`, SQL Server 2012 and later), transactions, `lastSerialID()` (identity columns, `SCOPE_IDENTITY()`), `escapeString()`, `subString()`, `concatString()` (`+` or `CONCAT()`), `md5()` (`HASHBYTES('MD5', ...)` converted to hex), `bitAnd()`/`bitOr()` (the `&` and `|` operators), table and sequence lists for `eZTableList()`
The `BIT_OR` aggregate	the driver	SQL Server has no `BIT_OR` aggregate; the content engine folds language masks with it. The SQLite driver solved the same gap by registering an aggregate; on SQL Server it needs a rewrite of those queries or a CLR aggregate
`UPDATE ... JOIN`	the driver	SQL Server writes it as `UPDATE t SET ... FROM t JOIN ...`; the SQLite driver's `rewriteJoinedUpdate()` shows the pattern
A schema handler	`lib/ezdbschema/classes/` and `settings/dbschema.ini`	turn `.dba` into `CREATE TABLE` with SQL Server types (`NVARCHAR`, `NVARCHAR(MAX)`, `INT IDENTITY`), indexes (within SQL Server's index key size limits), and back again for the upgrade check
SQL files	`kernel/sql/<engine>/`, `update/database/<engine>/`	the generated schema and the update files of every later release

Piece	Where	What it involves
Installer entries	`eZSetupDatabaseMap()` in `kernel/setup/ezsetupcommon.php`, `settings/setup.ini [database_extensions]`, `bin/php/install.php`	the wizard choice, the extension check, the `--db` alias
Session statements	the driver	ignore MySQL-only `SET` statements, as the SQLite driver does
Case and collation	the database	SQL Server's default collations compare without case, like MySQL's; choose a UTF-8 collation (SQL Server 2019 and later) or `NVARCHAR` columns
Extensions	every extension with its own SQL	the same review the PostgreSQL and Oracle ports needed: backticks, `LIMIT`, MySQL functions
Tests and CI	`tests/`, `.github/workflows/`	a CI job with a SQL Server container, as the MySQL and PostgreSQL jobs have

How large that work is has not been measured, and no estimate is given here. Anyone who wants to take it on is welcome: start from xrowmssql for the SQL Server specifics and from the `ezoracle` extension as the model of a current driver shipped outside the kernel, and open an issue on the software tracker to coordinate.

Character sets and collations

Exponential stores and serves text as UTF-8. Each engine names that differently:

Database	Create the database with	Notes
1. SQLite 3	nothing to choose	SQLite stores text as UTF-8. = compares with case; `LIKE` ignores case for ASCII letters only.
2. PostgreSQL	`--encoding=UTF8 --template=template0`	`template0` lets you choose the encoding regardless of the server's default. Comparisons are case-sensitive.
3. MongoDB	nothing to choose	BSON strings are UTF-8. Comparisons are case-sensitive unless a collation is set on the query or index.
4. MariaDB	`CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci`	The wizard refuses a database whose default character set is not UTF-8 ("cannot be used, the setup wizard wants to create the site in ..."). Default collations compare without case.
5. Oracle	database character set `AL32UTF8`	The driver connects in AL32UTF8; `NLS_LANG` is not needed. Case-sensitive unless `OracleCaseInsensitive=enabled`.
6. MySQL	`CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci`	As MariaDB.

The MySQL and MariaDB connection. `Charset=utf-8` (the default) is sent to the server as `utf8`, which both servers treat as the three-byte `utf8mb3`. Keep the database in `utf8mb4` anyway: it costs nothing. If the connection character set cannot be set, the driver keeps running and logs the warning "could not set the connection charset"; it is logged only when warnings are (`site.ini [DebugSettings] AlwaysLog[]=warning`).

Case. A login or identifier lookup written against MySQL's case-insensitive collations may find nothing on PostgreSQL, Oracle or MongoDB. Compare `lower(...)` on both sides where case must not matter.

The System information page (Setup > System information) warns when the connection's character set is not UTF-8.

Performance and scaling

- **Most requests never reach the database.** Velocity's response cache (anonymous visitors), the content view cache and the template-block cache answer first; below them the optional SQL query cache (`settings/querycache.ini`, `Mode=off` by default) answers repeated `SELECT` statements on every engine except MongoDB. Measure with `php bin/php/console exp:benchmark`.
- **SQLite** scales reads with the number of processes; writes queue. Count up to 64 MB of page cache per process that has opened the database (`SQLitePragmas[]=cache_size=-16384` lowers it to 16 MB).
- **PostgreSQL, MariaDB, MySQL** scale writes with row-level locking. `UsePersistentConnection=enabled` keeps one connection per PHP process; under Velocity's persistent workers each worker keeps its connection anyway. MySQL and MariaDB can send reads to replicas (`UseSlaveServer`).
- **MongoDB**: performance follows the number of statements per page; create the indexes from `bin/mongodb/create_indexes.js`, without which subtree queries and URL alias lookups scan whole collections.
- **Oracle**: a new dedicated session costs tens of milliseconds; use `Persistent=enabled` and DRCP (`ezoracle.ini [ConnectionSettings]`), and the `Prefetch` settings of `[PerformanceSettings]`.
- **Slow statements**: `[DatabaseSettings] SlowQueriesOutput=<milliseconds>` with `SQLOutput=enabled` lists the slower statements in the debug output on every engine; on MySQL and MariaDB `QueryAnalysisOutput=enabled` adds `EXPLAIN`; Oracle logs slow statements to `var/log/oracle-slow.log` when `[LogSettings] SlowQueryThreshold` is set.

The comparison table of 9.7 lists concurrency, replicas and persistent connections side by side.

Backup and restore

A complete backup is the database **and** the files: `var/storage` (uploads and image variations), `settings/override` and `settings/siteaccess`. Switch on maintenance mode for a restore (`php bin/php/maintenance.php on --allow-root-user`), and clear every cache afterwards (`php bin/php/ezcache.php --clear-all --allow-root-user` and, with Velocity, `php bin/php/console exp:velocity cache clear --apply --allow-root-user`). The full procedure is in chapter 10, backups and restore.

Database	Backup	Restore
1. SQLite 3	`sqlite3 var/storage/sqlite3/exponential.db ".backup '/backup/exponential.db'"` (online, consistent) or `"VACUUM INTO '/backup/exponential.db'"`; check with `PRAGMA integrity_check`	`sqlite3 var/storage/sqlite3/exponential.db ".restore '/backup/exponential.db'"`; never copy a database file that is in use
2. PostgreSQL	`pg_dump -U exponential -Fc exponential > exponential.dump`	`pg_restore -U exponential -d exponential --clean exponential.dump`

Database	Backup	Restore
3. MongoDB	`mongodump --uri="mongodb://exponential@localhost:27017/exponential" --out=/backup/exponential`	`mongorestore --uri="mongodb://exponential@localhost:27017/exponential" --drop /backup/exponential`
4. MariaDB	`mariadb-dump --single-transaction -u exponential -p exponential > exponential.sql`	`mariadb -u exponential -p exponential < exponential.sql`
5. Oracle	`php bin/php/console ext:ezoracle:datapump` (see its `--help`), or Oracle's `expdp`	Oracle's `impdp`
6. MySQL	`mysqldump --single-transaction -u exponential -p exponential > exponential.sql`	`mysql -u exponential -p exponential < exponential.sql`

`--single-transaction` gives a consistent dump of InnoDB tables without locking them. Commands that take a password on the command line show it to other users of the machine; prefer the tools' prompts or option files.

Moving a site to another database

The content stays the same when a site changes its database; only where it is stored changes. Plan a maintenance window: nothing may be published between the dump and the switch.

- **Between SQL engines** (SQLite, PostgreSQL, MariaDB, MySQL, and Oracle with `ezoracle`): dump with `bin/php/ezsqldumpschema.php --format=generic --output-array --output-types=all` into a `.dba` file and load it with `bin/php/ezsqlinsertschema.php --insert-types=all`, choosing the engine with `--type` (`mysql`, `postgresql`, `sqlite3`, `oracle`). Then point `DatabaseImplementation` and the connection keys at the new engine, clear all caches and restart the workers.
- **Into Oracle**: the extension's own `ext:ezoracle:mysql2oracle-schema` and `ext:ezoracle:mysql2oracle-data`.
- **Into MongoDB**: the scripts in `bin/mongodb/` (`export_mysql.sh`, `mysql2ndjson.py`, `import_all.sh`, `validate_ndjson.sh`) and section 18 of the porting reference.
- **Check before you reopen**: compare the row counts of `ezcontentobject`, `ezcontentobject_tree` and `ezcontentobject_attribute` on both sides; on PostgreSQL correct the sequences; sign in and publish a test object.
- **Later upgrades** use the update files of the engine the site runs on now, not the one it was installed on.

The step-by-step version is 9.9 Switching an existing site to another engine.

Test coverage (CI)

What the project's continuous integration (`.github/workflows/`) runs on every push, per database. "Driver unit tests" means tests that open a real database file or test the driver's SQL without a server; "database tests" means the legacy toolkit's tests run against a live server with a throw-away database per test.

Database	In CI	How
1. SQLite 3	yes, driver unit tests	the PHPUnit job on PHP 8.1, 8.2, 8.3, 8.4 and 8.5 loads `sqlite3`; the SQLite tests in `tests/tests/lib/ezdb/` check the driver, partly on an in-memory or temporary database (query counts, the schema mirror of `kernel/sql/sqlite/`) and partly by reading its settings (the transaction wait)
2. PostgreSQL	yes, database tests	the job "Database tests (postgresql, PHP 8.3)" with a `postgres:15` service and `pgcrypto`, `tests/runtests.php --dsn postgresql://... --db-per-test`
3. MongoDB	adapter unit tests	the PHPUnit job loads `mongodb` and installs `mongodb/mongodb`; the `mongodb` suite (adapter tests without a server) runs; the `mongodb-live` group needs running MongoDB and MySQL servers and is excluded from CI
4. MariaDB	no	covered indirectly by the MySQL job (same driver)
5. Oracle	no	Oracle-specific kernel tests skip on other engines; the `ezoracle` extension has no workflow of its own in the installed copy (not verified upstream)
6. MySQL	yes, database tests	the job "Database tests (mysql, PHP 8.3)" with a `mysql:8.0` service

PHP 8.0, the oldest supported version, and PHP 7.4 (usable without support) are checked by the Quality workflow with a syntax check of every PHP file and a boot smoke test without a database. The test suites and how to run them locally are described in Continuous integration.

Which database is my site using?

- In the administration: **Setup > System information**, the **Database** card: engine and server version, driver class, the file (SQLite) or server and database name, character set, number of tables and size.
- On the command line:

  Shell

  ```
  php bin/php/console exp:ini get site.ini/DatabaseSettings/DatabaseImplementation --allow-root-user
  ```

- **Setup > Upgrade check**, "Check database consistency", compares the tables, fields and indexes with the schema of the release for MySQL and MariaDB, PostgreSQL, SQLite and MongoDB (The upgrade check).
- In code: `eZDB::instance()->databaseName()` returns `sqlite`, `postgresql`, `mongo`, `mysql` or `oracle`.

Related pages

- Chapter 9, Databases: the operator's reference for every engine.
- Chapter 2, Requirements and OS support.
- SQLite database support, SQLite: transactions queue for the write lock, The SQLite3 database driver.
- MongoDB as the database and the MongoDB kernel support porting reference.
- SQLite and Oracle drivers, Database drivers and installers, September 2026.

- SQL query cache, Installing in one command, Kickstarter, System information.
- Free documentation: these pages are free to copy, translate and improve.

> **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.