SQLite maintenance: ANALYZE, VACUUM and the WAL file

The routine maintenance of an Exponential SQLite database - statistics with ANALYZE and PRAGMA optimize, compaction with VACUUM in maintenance mode, keeping the WAL file in check with journal_size_limit and a TRUNCATE...

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

This page is for database administrators who look after an Exponential site on SQLite. It answers: which maintenance SQLite needs, when to run each task, how to run it without disturbing editors, and what healthy output looks like. Source: ANALYZE, VACUUM and the WAL file and Monitoring a SQLite site. The `sqlite3` command-line program is optional for the site but needed for these tasks (`sqlite3 --version`).

Contents

1. Statistics: ANALYZE
2. Compaction: VACUUM
3. The WAL file
4. Integrity checks
5. A maintenance schedule
6. Checklist

The examples use `var/storage/sqlite3/exponential.db`; use the file your `Database` setting names.

Statistics: ANALYZE

SQLite's query planner chooses indexes better with statistics. Run `ANALYZE` once after an install or a large import, and again after the content has grown a lot:

Shell

```
sqlite3 var/storage/sqlite3/exponential.db "ANALYZE"
```

A site that is slow right after an install or an import usually lacks them (12.10). `PRAGMA optimize` runs only the analysis that is needed and is cheap enough for a weekly cron entry:

Shell

```
sqlite3 var/storage/sqlite3/exponential.db "PRAGMA optimize"
```

Compaction: VACUUM

Deleted content leaves free pages in the file. SQLite reuses them but does not give them back to the file system. `VACUUM` rebuilds the file compactly. It needs about twice the database's size in free disk space and holds the write lock for its whole run, so writers queue behind it and may

time out after `SQLiteTransactionWait`. Run it in maintenance mode, and only when the file has become much larger than its content (after removing a large part of the site):

Shell

```
php bin/php/maintenance.php on --allow-root-user
sqlite3 var/storage/sqlite3/exponential.db "VACUUM"
php bin/php/maintenance.php off --allow-root-user
```

`VACUUM INTO '<file>'` writes a compacted copy instead and is a good backup method (Backups per engine).

The WAL file

After a checkpoint SQLite reuses the WAL file from its start but does not shrink it. A burst of heavy writing can leave a large `-wal` file behind; in the project's load tests it grew well over a hundred megabytes and was checkpointed back afterwards. Two remedies:

- In `settings/override/site.ini.append.php`, `SQLitePragmas[]=journal_size_limit=67108864` truncates the WAL to 64 MB after each checkpoint that empties it.
- A manual checkpoint that truncates the file:

  Shell

  ```
  sqlite3 var/storage/sqlite3/exponential.db "PRAGMA wal_checkpoint(TRUNCATE)"
  ```

  It prints three numbers; a first number of `0` means the checkpoint was not blocked.

Never delete the `-wal` or `-shm` file while any process has the database open: they are part of the database.

Integrity checks

Check	Command	Healthy
Quick	`sqlite3 var/storage/sqlite3/exponential.db "PRAGMA quick_check"`	`ok`
Thorough	`sqlite3 var/storage/sqlite3/exponential.db "PRAGMA integrity_check"`	`ok`
Journal mode	`sqlite3 var/storage/sqlite3/exponential.db "PRAGMA journal_mode"`	`wal`
Disk space	`df -h var/storage/sqlite3`	room for the database twice over

Run the same integrity check on every backup before you rely on it.

A maintenance schedule

When	Task
After install, large import, engine move	`ANALYZE`
Weekly (cron, off-peak)	`PRAGMA optimize`; `PRAGMA quick_check`; check the `-wal` size

When	Task
After removing a large part of the site	`VACUUM` in maintenance mode
After a burst of writes left a large WAL	`PRAGMA wal_checkpoint(TRUNCATE)`, or set `journal_size_limit`
Daily	online backup and `PRAGMA integrity_check` of the backup

Run cron entries with the same group and a umask of `0002` as the site, so files they create stay group-writable (SQLite files, ownership and permissions). `exp:benchmark kernel` measures the database round trip among its probes; with `--save` and `--baseline` it catches a slowdown between two runs (Benchmark).

Checklist

- ☐ `ANALYZE` was run after the install and after large imports.
- ☐ A weekly `PRAGMA optimize` and `quick_check` are scheduled.
- ☐ `VACUUM` runs only in maintenance mode, with enough free disk.
- ☐ The `-wal` file does not keep growing after busy periods.

Related pages

- SQLite 3 for database administrators
- SQLite: the write queue and locking
- Monitoring the database
- Maintenance mode
- 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.