database upgrade

In this article

Upgrade a database so it is usable by the current tools.

GitHub CodeQL is licensed on a per-user basis upon installation. You can use CodeQL only for certain tasks under the license restrictions. For more information, see "About the CodeQL CLI." If you have a GitHub Advanced Security license, you can use CodeQL for automated analysis, continuous integration, and continuous delivery. For more information, see "About GitHub Advanced Security."

This content describes the most recent release of the CodeQL CLI. For more information about this release, see https://github.com/github/codeql-cli-binaries/releases.

To see details of the options available for this command in an earlier release, run the command with the `--help` option in your terminal.

Synopsis

Shell

```
codeql database upgrade [--threads=<num>] [--ram=<MB>] <options>... -- <database>
```

Description

Upgrade a database so it is usable by the current tools.

This rewrites a CodeQL database to be compatible with the QL libraries that are found on the QL pack search path, if necessary.

If an upgrade is necessary, it is irreversible. The database will subsequently be unusable with the libraries that were current when it was created.

Options

Primary Options

`<database>`

[Mandatory] Path to the CodeQL database to upgrade.

`--search-path=<dir>[:<dir>...]`

A list of directories under which QL packs containing upgrade recipes may be found. Each directory can either be a QL pack (or bundle of packs containing a `.codeqlmanifest.json` file at the root) or the immediate parent of one or more such directories.

If the path contains directories trees, their order defines precedence between them: if a pack name that must be resolved is matched in more than one of the directory trees, the one given first wins.

Pointing this at a checkout of the open-source CodeQL repository ought to work when querying one of the languages that live there.

(Note: On Windows the path separator is `;`).

`--additional-packs=<dir>[:<dir>...]`

[Advanced] If this list of directories is given, they will be searched for upgrades before the ones in `--search-path`. The order between these doesn't matter; it is an error if a pack name is found in two different places through this list.

This is useful if you're temporarily developing a new version of a pack that also appears in the default path. On the other hand it is *not recommended* to override this option in a config file; some internal actions will add this option on the fly, overriding any configured value.

(Note: On Windows the path separator is `;`).

`--target-dbscheme=<file>`

The *target* dbscheme we want to upgrade to. If this is not given, a maximal upgrade path will be constructed

`--target-sha=<sha>`

[Advanced] An alternative to `--target-dbscheme` that gives the internal hash of the target dbscheme instead of the dbscheme file.

`--[no-]allow-downgrades`

Include any relevant downgrades if there are no upgrades

Options to control evaluation of upgrade queries

`--[no-]tuple-counting`

[Advanced] Display tuple counts for each evaluation step in the query evaluator logs. If the `--evaluator-log` option is provided, tuple counts will be included in both the text-based and structured JSON logs produced by the command. (This can be useful for performance optimization of complex QL code).

`--timeout=<seconds>`

[Advanced] Set the timeout length for query evaluation, in seconds.

The timeout feature is intended to catch cases where a complex query would take "forever" to evaluate. It is not an effective way to limit the total amount of time the query evaluation can take. The evaluation will be allowed to continue as long as each

separately timed part of the computation completes within the timeout. Currently these separately timed parts are "RA layers" of the optimized query, but that might change in the future.

If no timeout is specified, or is given as 0, no timeout will be set (except for codeql test run, where the default timeout is 5 minutes).

`-j, --threads=<num>`

Use this many threads to evaluate queries.

Defaults to 1. You can pass 0 to use one thread per core on the machine, or -*N* to leave *N* cores unused (except still use at least one thread).

`--[no-]save-cache`

[Advanced] Aggressively write intermediate results to the disk cache. This takes more time and uses (much) more disk space, but may speed up the subsequent execution of similar queries.

`--[no-]expect-discarded-cache`

[Advanced] Make decisions about which predicates to evaluate, and what to write to the disk cache, based on the assumption that the cache will be discarded after the queries have been executed.

`--[no-]keep-full-cache`

[Advanced] Don't clean up the disk cache after evaluation completes. This may save time if you're going to do codeql dataset cleanup or codeql database cleanup afterwards anyway.

`--max-disk-cache=<MB>`

Set the maximum amount of space that the disk cache for intermediate query results can use.

If this size is not configured explicitly, the evaluator will try to use a "reasonable" amount of cache space, based on the size of the dataset and the complexity of the queries. Explicitly setting a higher limit than this default usage will enable additional caching which can speed up later queries.

`--min-disk-free=<MB>`

[Advanced] Set target amount of free space on file system.

If `--max-disk-cache` is not given, the evaluator will try hard to curtail disk cache usage if the free space on the file system drops below this value.

`--min-disk-free-pct=<pct>`

[Advanced] Set target fraction of free space on file system.

If `--max-disk-cache` is not given, the evaluator will try hard to curtail disk cache usage if the free space on the file system drops below this percentage.

`--external=<pred>=<file.csv>`

A CSV file that contains rows for external predicate *<pred>*. Multiple `--external` options

can be supplied.

`--xterm-progress=<mode>`

[Advanced] Controls whether to show progress tracking during QL evaluation using xterm control sequences. Possible values are:

`no`: Never produce fancy progress; assume a dumb terminal.

`auto` *(default)*: Autodetect whether the command is running in an appropriate terminal.

`yes`: Assume the terminal can understand xterm control sequences. The feature still depends on being able to autodetect the *size* of the terminal, and will also be disabled if `-q` is given.

`25x80` (or similar): Like `yes`, and also explicitly give the size of the terminal.

`25x80:/dev/pts/17` (or similar): show fancy progress on a *different* terminal than stderr. Mostly useful for internal testing.

Options for controlling outputting of structured evaluator logs

`--evaluator-log=<file>`

[Advanced] Output structured logs about evaluator performance to the given file. The format of this log file is subject to change with no notice, but will be a stream of JSON objects separated by either two newline characters (by default) or one if the `--evaluator-log-minify` option is passed. Please use `codeql generate log-summary <file>` to produce a more stable summary of this file, and avoid parsing the file directly. The file will be overwritten if it already exists.

`--evaluator-log-minify`

[Advanced] If the `--evaluator-log` option is passed, also passing this option will minimize the size of the JSON log produced, at the expense of making it much less human readable.

Options to control RAM usage of the upgrade process

`-M, --ram=<MB>`

Set total amount of RAM the query evaluator should be allowed to use.

Common options

`-h, --help`

Show this help text.

`-J=<opt>`

[Advanced] Give option to the JVM running the command.

(Beware that options containing spaces will not be handled correctly.)

`-v, --verbose`

Incrementally increase the number of progress messages printed.

`-q, --quiet`

Incrementally decrease the number of progress messages printed.

`--verbosity=<level>`

[Advanced] Explicitly set the verbosity level to one of errors, warnings, progress, progress+, progress++, progress+++. Overrides `-v` and `-q`.

`--logdir=<dir>`

[Advanced] Write detailed logs to one or more files in the given directory, with generated names that include timestamps and the name of the running subcommand.

(To write a log file with a name you have full control over, instead give `--log-to-stderr` and redirect stderr as desired.)