Fixing your database schema

Christopher Nadeau - 2019-01-11 - Comments (0) - Deskpro Legacy

Q: What is the schema?

The *database schema* is the layout of the database that stores all information in Deskpro. The schema describes how data is organised and the rules or constraints on that data. For example, it describes that tickets have a numeric ID and ticket messages contain text, and that messages are linked to tickets.

In rare cases, your database schema might differ from what the Deskpro engineers intend it to be. For example, imagine if the link between messages and tickets was not enforced -- you could easily end up with orphan messages.

While problems like that are extreme and rare, we always recommend you run the automated maintenance tool if the system ever detects a schema difference just so you can be sure.

Q: I got an alert that my schema was different

Deskpro will automatically scan your database every day to verify your schema. If your schema differs, you may see a yellow notice in your admin interface or a warning before you run an upgrade.

Most of the time, schema differences are very trivial and can be completely ignored. But sometimes schema differences might be an indication of data corruption -- and this is why the warning appears.

The warning is an indication that there *might* be an issue. But in *most* cases, you shouldn't worry. You can just run the maintenance tool described below to fix any minor discrepancies and go on with your day.

Automated schema maintenance

Before you begin: Always backup your database before performing any kind of database work, including the steps described here. You can find instructions on how to backup your database here:
https://support.deskpro.com/en/guides/sysadmin-guide/backups/mysql-backup

If at any point you are unsure or need assistance, submit a ticket and our technicians will be happy to help.

Fixing your schema

Deskpro includes a command-line tool designed to help identify and fix schema problems. Here's an example usage to fix all possible errors:

```
$ bin/console dp:update:fix-schema --fix-all

!!! NOTE !!!

You have not specified the --run flag, so no changes are being
applied. This is running in PREVIEW mode.
Fixing foreign keys ...
ALTER TABLE email_sources ADD CONSTRAINT FK_6F9D0D3DED3E8EA5 FOREIGN
KEY (blob_id) REFERENCES blobs (id) ON DELETE CASCADE
Done fixing foreign keys.

All done.
```

Running the tool always runs in *preview mode* unless you specify the --run flag. You'll see a summary of the errors the tool detected. This gives you a chance to review the proposed changes before actually applying them. When you're ready to actually apply the fixes, supply the --run flag.

```
$ bin/console dp:update:fix-schema --fix-all --run
Fixing foreign keys ...
ALTER TABLE email_sources ADD CONSTRAINT FK_6F9D0D3DED3E8EA5 FOREIGN
KEY (blob_id) REFERENCES blobs (id) ON DELETE CASCADE
Done fixing foreign keys.
```

```
All done.
```

Options

You can specify specific things to look for and fix. --fix-all runs everything *except* --ref-integrity.

--fix-tables	This checks all tables to ensure they exist, all expected fields exist, and they are the correct type.
--fix-indexes	This checks for indexes.
--fix-fks	This checks for foreign key constraints. This is perhaps the most important thing that needs to be correct!
--ref-integrity	This checks every table and verifies foreign key referential integrity. Foreign key constraints enforce, at the database level, that a relationship from one table maps to a valid record in another. For example, a ticket belongs to a user. A FK relationship makes the database enforce that the user stored on a ticket actually exists. The database makes it impossible to set an invalid value, such as the ID of a user that doesn't exist. If your database is missing foreign key constraints (i.e. --fix-fks above found problems), then it's possible that the database is corrupted and relationships contain invalid data. Using --fix-ref-integrity scans every table and every FK constraint and validates the records are correct. If they are incorrect, then the tool will attempt to fix the problem by setting the value to NULL or by deleting the offending record. Using --fix-ref-integrity can take a very long time, depending on the size of your database. The system needs to iterate over every single row in every single table and check every single column with a foreign key. Note also that this may also be destructive. Sometimes data may be missing such that the only way to restore full integrity is to delete a row containing the invalid reference. This is why making a backup is critical before running the tool.

Referential integrity

As noted above, the --ref-integrity option can take a very very long time on larger databases. This step is often unnecessary and can usually be skipped or cancelled.

As a rule-of-thumb: If you haven't noticed any odd behaviour in Deskpro and aren't getting logged errors about missing database rows, you can skip the integrity check.

If the ref check process has already begun, you can safely cancel it by stopping the process (i.e. close your terminal window or press CTRL+C). Just note that if you re-run the process, it will re-start form the beginning.

If you do want to run the ref check in preview mode, note that **it is safe to leave running in the background**. You can keep the ref checks running even while Deskpro is in operational in production. This will have a small impact in performance (the ref check is doing thousands of SELECT queries), but generally it will not impact your agents.

However, note that when fixing ref integrity problems (i.e. you're running with the --run flag), this *will make changes to your database records*. So it's recommended to only --run on a disabled helpdesk just in case you need to recover to a backup.

To run the referential integrity check:

```
$ bin/console dp:update:fix-schema --ref-integrity --run
```

Q: How might the schema stray from the default?

There are a number of reasons your schema might be different from the default. The most common reasons:

Upgrading from older versions

Some older versions of Deskpro might use different machine names for indexes or keys. For example, even if you have an index on a column, it might be called "foo" while the system thinks it should be called "foo_idx". In a case like this, the only difference is the name -- there is no functional difference to how the system actually operates. These sorts of problems are typically very trivial and easily solved.

Another reason might be upgrades that failed in the past. If you've ever had an issue with

an upgrade, it's possible that a certain upgrade step was skipped or didn't complete fully. That means past migrations (scripts that change/update the schema) never got applied to your database.

User-error (i.e. sysadmin mistakes)

More serious corruption is usually caused by user-error. For example, we've seen cases where a sysadmin is moving a database from one server to another but failed to export tables with all the appropriate keys or indexes. If this goes unnoticed, this very quickly begins to introduce very serious issues that ultimately lead to failures and errors while using Deskpro.