Operating PostgreSQL: sequences, case and gotchas

Day-to-day knowledge for running Exponential on PostgreSQL - how auto-increment columns map to sequences named table_column_seq, how to correct them after loading data, the sequence rename that sites from 5.x need...

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

This page is for database administrators who run an Exponential site on PostgreSQL. It answers: how the schema handles auto-increment ids, what to do after a data load or a partial restore, what changed in sequence names between 5.x and 6.0, how comparisons differ from MySQL, and which errors are common. Sources: 9.4 PostgreSQL and 11.3 The 6.12, 7.2 and 7.3 directories.

Contents

1. Sequences
2. Correcting sequences
3. Sequence names on sites from 5.x
4. Case and comparison
5. Persistent connections
6. Backups
7. Gotchas

Sequences

Auto-increment columns are integers with a default of `nextval('<table>_<column>_seq')`. The schema handler (`lib/ezdbschema/classes/ezpgsqlschema.php`) creates each sequence before its table (`CREATE SEQUENCE IF NOT EXISTS`), and the driver's `lastSerialID()` reads `currval('<table>_<column>_seq')` after an insert. For example, the ids of `ezcontentobject` come from `ezcontentobject_id_seq`.

Correcting sequences

When data is loaded with explicit ids (an import, a restore of selected tables, a move from another engine), the sequences stay behind and the next insert fails with a duplicate key. The schema installer corrects them itself after it loads data (`correctSequenceValues()`); for one table by hand:

SQL

```
SELECT setval('ezcontentobject_id_seq', (SELECT max(id) FROM ezcontentobject));
```

The kernel ships `kernel/sql/postgresql/setval.sql` with such statements for the kernel's

tables. Read it before you run it, and run it as the site's account.

Sequence names on sites from 5.x

Up to 5.4 the sequences were named `<table>_s`. The 6.0 kernel reads `<table>_<column>_seq`, so on a database that still has the old names every insert that needs its new id fails. The 5.4 to 6.0.0 update file (`update/database/postgresql/6.0/dbupdate-5.4-to-6.0.sql`) renames them, each only when the old name exists and the new one is free, and points each column default at the new name; the 6.0.0 to 6.0.15 file does the same again for a site that reached 6.0 with an older copy. Its guarded steps are `DO` blocks.

Do not apply `update/database/postgresql/7.2/dbupdate-6.13.0-to-7.2.0.sql` as a file: it comes from the upstream kernel of 2018, starts from a version no Exponential database has, and its change is already in the 6.0 files.

Check with `\ds` in `psql`: the list shows `<table>_<column>_seq` names only.

Case and comparison

PostgreSQL compares strings with case. MySQL's default collations do not. A login or identifier lookup written for MySQL (in an extension, or in your own report query) may therefore find nothing on PostgreSQL. Compare `lower(...)` on both sides where case must not matter (Character sets and collations).

Persistent connections

`UsePersistentConnection=enabled` makes the driver use `pg_pconnect` instead of `pg_connect`, keeping one connection per PHP process. Under Velocity's persistent workers each worker keeps its connection for its whole life anyway. Count the connections you need: PHP-FPM workers plus Velocity workers plus cron, against `max_connections` on the server. A connection pooler in front of PostgreSQL is a server-side choice; Exponential has no setting of its own for one, so test it on staging before production.

Backups

Shell

```
pg_dump -U exponential -Fc exponential > exponential.dump
pg_restore -U exponential -d exponential --clean exponential.dump
```

The custom format (`-Fc`) is compressed and restores selectively. Point-in-time recovery and streaming replication are PostgreSQL's own features and work with Exponential as with any client; configure them on the server (Backups per engine).

Gotchas

Symptom	Cause
"The 'digest' function is not available in your database ..."	`pgcrypto` is missing; create it (Setting up PostgreSQL)
Connection refused	`listen_addresses` or `pg_hba.conf`

Symptom	Cause
Duplicate key on insert after an import	Sequences behind their tables; correct them as above
Every insert fails after an upgrade from 5.x	Old `<table>_s` sequence names; apply the current 6.0 update file
An extension's SQL fails on PostgreSQL only	MySQL syntax such as backtick quoting or MySQL-only SET statements (Reviewing extension SQL)
A login works only in one spelling	Case-sensitive comparison

PostgreSQL's `errorNumber()` in Exponential is the SQLSTATE (a string such as `'23505'` for a duplicate key); before 6.0.15 it was `1` for every error (Database error codes and messages).

Related pages

- Setting up PostgreSQL
- Upgrades and the SQL update files
- PostgreSQL for hosts
- 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.