Everything you need to know about collations

Andreas Karlsson

Percona

Who am I?

- Andreas Karlsson
- Software engineer for 18 years
- Minor PostgreSQL contributor for 14 of them
- Works at Percona with `pg_tde`, `pg_oidc_validator` and PostgreSQL

PERCONA

Does this look familiar?

```
WARNING:  database "foo" has a collation version mismatch
DETAIL:  The database was created using collation version 2.38, but the operating system
provides version 2.30.
HINT:  Rebuild all objects in this database that use the default collation and run ALTER
DATABASE foo REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.
```

Agenda

1. What are collations?
2. Version missmatch problem
3. Using collations
4. `libc` provider
5. `builtin` provider
6. `icu` provider
7. My oppinions

What are collations?

PERCONA

Merriam-Webster

collate verb

1. a. to compare critically
   b. to collect, compare carefully in order to verify, and often to integrate or arrange in order
   "collated the data for publication"
2. a. **to assemble in proper order**
   ***especially*: to assemble in order for binding**
   "collate printed sheets"
   b. to verify the order of (printed sheets)
3. [Latin *collatus*, past participle]: to institute (a cleric) to a benefice

PERCONA

Why do we care about it?

What if we just sort text in binary or unicode order?

```
$ SELECT * FROM cities ORDER BY city COLLATE pg_c_utf8;
   city
-----------
 Aarhus
 Essen
 Oslo
 Valencia
 Ängelholm
 Åre
 Öhringen
 Östersund
(8 rows)
```

Why do we care about it?

Maybe more what most people expect.

```
SELECT * FROM cities ORDER BY city COLLATE "de-DE-x-icu";
   city
-----------
 Aarhus
 Ängelholm
 Åre
 Essen
 Öhringen
 Oslo
 Östersund
 Valencia
(8 rows)
```

PERCONA

Why do we care about it?

Swedish albhabet is ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖ.

```
SELECT * FROM cities ORDER BY city COLLATE "sv-SE-x-icu";
   city
-----------
 Aarhus
 Essen
 Oslo
 Valencia
 Åre
 Ängelholm
 Öhringen
 Östersund
(8 rows)
```

Why do we care about it?

Danish albhabet is ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÅ.

```
SELECT * FROM cities ORDER BY city COLLATE "da-DK-x-icu";
   city
-----------
 Essen
 Oslo
 Valencia
 Ängelholm
 Öhringen
 Östersund
 Åre
 Aarhus
(8 rows)
```

What are they in PostgreSQL?

- Part of the SQL standard
- Defines which order to sort text types: `text`, `varchar`, `char`
- Controls `ORDER BY` and comparison opeartors for text types
- Also controls `upper()`, `lower()`, `initcap()`, `casefold()`

The version problem

PERCONA

Why does this happen?

```
WARNING:  database "foo" has a collation version mismatch
DETAIL:  The database was created using collation version 2.38, but the operating system
provides version 2.30.
HINT:  Rebuild all objects in this database that use the default collation and run ALTER
DATABASE foo REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.
```

- Default in PostgreSQL is to use libc and the current locale set on initdb
- Collation order changes between
- Collation order changes can corrupt indexes

Why can we get index corruption?

- B-tree are stored on disk in sorted order

How to fix it

- Folow the instructions
  - Pray your indexes are small
  - Reindex everything **or** try to find a query to find only affected indexes (most are wrong/misleading, including the one in our docs)
  - Run `ALTER DATABASE ... REFRESH COLLATION VERSION`
- Manually figure out which indexes need to be reindexed
  - Check for ASCII-only fields
  - Check which collations actually changed

    Jeremey Schnedier: https://github.com/ardentperf/glibc-unicode-sorting
- Use logical replication
- Any way to prevent it?

PERCONA

Using collations

PERCONA

Using collation

- Controls `ORDER BY` and comparison opeartors
- Also controls `upper()`, `lower()`, `initcap()`, `casefold()`
- Providers
  - libc
  - ICU (PostgreSQL 10)
  - builtin (PostgreSQL 17)

What can have a collation?

- Cluster has a default collation
- Databases have a default collation
  - Make sure to use `template0` if it is different from cluster collation
- Text columns have a colation
- Domain types may have a collation
- Fields of record types may have a collation
- Expressions may have a collation

How do collations for expressions work?

Sample table using ICU because I like it.

```
CREATE TABLE test (
  letter text,
  de text COLLATE "de-DE-x-icu",
  en text COLLATE "en-US-x-icu"
);

INSERT INTO test (letter, de, en) VALUES ('b', 'Bier', 'beer');
```

How do collations for expressions work?

Comparisons will be done uisng the collation of the column by default, in this case `de-DE-x-icu`.

```
SELECT * FROM test WHERE de > 'foo';
SELECT * FROM test ORDER BY de;
SELECT upper(de) FROM test;
```

How do collations for expressions work?

You can override the collation with an explicit one. Remomber thought that there may not be any index.

```
SELECT * FROM test WHERE de COLLATE "en-US-x-icu" > 'foo';
SELECT * FROM test ORDER BY de COLLATE "en-US-x-icu" ;
SELECT upper(de COLLATE "en-US-x-icu" ) FROM test;
```

How do collations for expressions work?

Explicit collations take presdence of implicit ones from e.g. columns.

```
SELECT * FROM test WHERE de COLLATE "fr-FR-x-icu" > en;
```

How do collations for expressions work?

If there are conflicting implicit or explicit we will get an error.

```
SELECT * FROM test WHERE de > en;
```

```
ERROR:  could not determine which collation to use for string comparison
HINT:  Use the COLLATE clause to set the collation explicitly.
```

```
SELECT * FROM test WHERE de COLLATE "fr-FR-x-icu" > en COLLATE "da-DK-x-icu";
```

```
ERROR:  collation mismatch between explicit collations "fr-FR-x-icu" and "da-DK-x-icu"
```

How do collations for expressions work?

If there are multiple collations as input to a function or operator we will get an column with an unknown collation.

```
SELECT * FROM test WHERE de > de || en; -- fails
```

A useful debugging tool when you do not know is `pg_collation_for()`. It is similar to `pg_typeof()`.

```
SELECT pg_collation_for(letter) letter, pg_collation_for(en) en, pg_collation_for(de) de,
pg_collation_for(en || de) ende, pg_collation_for(en || 'foo') foo FROM test;
  letter   |      en       |      de       | ende |      foo
-----------+---------------+---------------+------+--------------
 "default" | "en-US-x-icu" | "de-DE-x-icu" |      | "en-US-x-icu"
(1 row)
```

PERCONA

PROVIDER = libc

`PROVIDER = libc`

- Default collation provider
- Uses your libc
- Depends on your platform
- Requires reindex on libc upgrades
- Need to install libc locales
- Typically slow

The `C` locale

- The `C/POSIX` locale does not actually use libc
- Stable across versions
- Only ascii support for `upper()`, `lower()`, `initcap()`, `casefold()`

A way to avoid upgrade problems

- `compat-collation-for-glibc`
- Rips out collation code from glibc
- Allows you to use an old version
- https://github.com/awslabs/compat-collation-for-glibc

PROVIDER = builtin

PERCONA

`PROVIDER = builtin`

- Same across all platforms
- Does not require reindex on upgrades
- `pg_c_utf8` and `pg_unicode_fast`
- Fast
- UTF-8 only
- Added in PostgreSQL 17

PROVIDER = icu

`PROVIDER = icu`

- Uses ICU4C from the Unicode Consotrium
- Requires reindex on libc upgrades
- Ships with > 800 collations by default
- Provides `und/unicode` collation which is “universal”
- Generally faster than your libc, speed improvements in PostgreSQL 19 for UTF-8
- Allows customizable collations:
  - BCP 47
  - LDML rule syntax
- Added in PostgreSQL 10

BCP 47

- Example `de-DE-u-kn-level1`
- Decide on type: e.g. normal vs phonebook
- Ordering of punctuation or ignoring punctuation
- Handle case sensitivity
- Control how accented characters are treated
- Select numeric or lexical sorting of numbers
- Enable unicode noramlization

BCP 47

RFC for specifying language preferences including collations.

```
-- ignore differences in accents and case
CREATE COLLATION ignore_accent_case (provider = icu, deterministic = false, locale = 'und-u-
ks-level1');
SELECT 'Å' = 'A' COLLATE ignore_accent_case; -- true
SELECT 'z' = 'Z' COLLATE ignore_accent_case; -- true

-- upper case letters sort before lower case.
CREATE COLLATION upper_first (provider = icu, locale = 'und-u-kf-upper');
SELECT 'B' < 'b' COLLATE upper_first; -- true

-- treat digits numerically and ignore punctuation
CREATE COLLATION num_ignore_punct (provider = icu, deterministic = false, locale = 'und-u-
ka-shifted-kn');
SELECT 'id-45' < 'id-123' COLLATE num_ignore_punct; -- true
SELECT 'w;x*y-z' = 'wxyz' COLLATE num_ignore_punct; -- true
```

PERCONA

ICU Tailoring Rules / LDML rule syntax

Documented at: https://unicode-org.github.io/icu/userguide/collation/customization/

```
CREATE COLLATION ebcdic (provider = icu, locale = 'und',
rules = $$
& ' ' < '.' < '<' < '(' < '+' < \|
< '&' < '!' < '$' < '*' < ')' < ';'
< '-' < '/' < ',' < '%' < '_' < '>' < '?'
< '`' < ':' < '#' < '@' < \' < '=' < '"'
<*a-r < '~' <*s-z < '^' < '[' < ']'
< '{' <*A-I < '}' <*J-R < '\' <*S-Z <*0-9
$$);

SELECT c
FROM (VALUES ('a'), ('b'), ('A'), ('B'), ('1'), ('2'), ('!'), ('^')) AS x(c)
ORDER BY c COLLATE ebcdic;
 c
---
 !
```

PERCONA

PROVIDER = icu


```
a
b
^
A
B
1
2
```

PERCONA

My oppinions

My oppinions

- Use `pg_c_utf8` by default
- Use ICU for everything where natural language order matters
- Never use glibc locales
- The `C` locale for libc should have been a builtin locale

Questions?