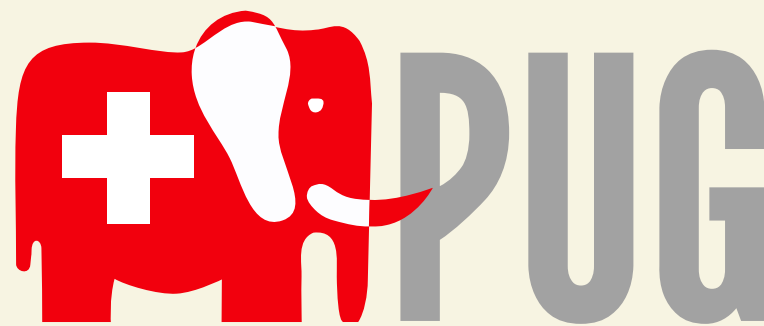


COLLATIONS IN POSTGRESQL: THE GOOD, THE BAD AND THE UGLY.

Swiss PGDay 2022, Rapperswil



Tobias Bussmann

2022-07-01

OVERVIEW

- What are collations?
- *the good*: practical use for developers
- *the bad*: things to watch out for
- *the ugly*: avoid data corruption
- History / Future

WHAT ARE COLLATIONS?

- *i18n* (internationalisation) feature
- part of *locale*
- related to *encoding*

LOCALE

Set of parameters to tell an application how users would expect it's output (and behaviour).

On `*NIX` traditionally set by environment variables:

- `LANG`: default main 'switch' (preferred)
- `LC_ALL`: for temp. override
- `LC_MESSAGES`: Language of user interface, `LANGUAGE` for a preference list
- `LC_TIME`: format of date and time
- `LC_NUMERIC`: decimal delimiter, grouping
- `LC_MONETARY`: Currency format and symbols
- `LC_CTYPE`: Character classification, case folding
- `LC_COLLATE`: string collation rules
- ...

Applied: `(LANGUAGE) > LC_ALL > LC_* > LANG`

COLLATIONS

- order of characters
- like learnt in elementary school
- simple but depending on culture

HOW ARE COLLATIONS RELEVANT?

- ORDER BY: order of output
- WHERE: searching data
- JOIN: combining data
- UNIQUE: define a sense of equality
- PARTITION BY: distributing data

Often supported by Indexes.

PROVIDERS

Functionality provided by the OS Different implementations

- POSIX: `libc` (Linux: `glibc`, `uClibc`; *BSD `libc`;
Windows: `msvcrt`; macOS: `libSystem`)
must match encoding
- ICU: common, portable library
supported for most encodings, mainly UTF8

ICU: pg10

<code>libc</code>	ICU
<code>strcoll</code>	<code>ucol_strcoll</code>
<code>strxfrm</code>	<code>ucol_getSortKey</code>
...	...

strcmp

decide if one string is *smaller*, *equal* or *greater* than a second one

- `<0`: string1 less than string2
- `0`: string1 identical to string2
- `>0`: string1 greater than string2

UNICODE NORMAL FORMS

Unicode: ~~Glyphs~~ Graphemes, Codepoints, Encodings

- composition
 - NFC/NFKC: composed form: Ä (U+00C4)
 - NFD/NFKD: decomposed form A+ö (U+0041 U+0308)
- equivalent (NFC / NFD) vs. compatible (NFKC / NFKD)
 - equal glyphs, meaning
 - variants, formatting, functions

UTR #15: UNICODE NORMALIZATION FORMS Full chart

THE GOOD

powerful support in PostgreSQL

NORMAL FORMS IN POSTGRESQL

- check

```
text IS [NOT] [form] NORMALIZED → boolean
```

- convert


```
normalize( text [, form ] ) → text
```

`form` is key word: NFC (default), NFD, NFKC, or NFKD

pg13, faster in pg14

NON-DETERMINISTIC COLLATION

Depending on the use case differences may be irrelevant:

- Case: a = A
- Normal form a+ = ä
- Accent: ä = a
- Phonebook: ä = ae

By default if `strcoll = 0`, `strcmp` is used as a tie-breaker, unless the collation is not defined `non-deterministic`.

pg12

COLLATE CLAUSE

- per **Expression**

```
WHERE a < b COLLATE "C"  
ORDER BY city COLLATE "de_CH"
```

- per **Column or Index**

```
CREATE TABLE t (c TEXT COLLATE "en_US")  
CREATE INDEX ON t (c COLLATE "cs_CZ")
```

- per **Domain** (and Composite and Range Types)

Default collatable types: TEXT, VARCHAR, CHAR

pg9.1

ORDER OF PRECEDENCE

1. explicit in expression
2. from column / domain
3. database default

Must be unambiguous, but 2. can mix collations in case the operator does not require a collation (|| vs. >)

Useful for testing:

`COLLATION FOR (<expression>)` can return NULL if undefined / mixed

pg9.2

DEFAULT COLLATIONS

Collation not configurable in Session or Config, no GUCs like `datestyle`

- compile time: `--with-icu`
- cluster creation: inherited from environment or set explicitly

```
initdb --encoding= --locale= --lc-collate= --lc-ctype=
```

- database creation: inherited from template

```
CREATE DATABASE name [ENCODING [=] encoding] [LC_COLLATE [=]  
lc_collate] [LC_CTYPE [=] lc_ctype];
```

global: pg6.1, DB level: pg8.4, ICU: pg10

where is ICU on cluster/db level?

ICU AS DEFAULT COLLATION

- cluster creation:

```
initdb --locale-provider=icu --icu-locale=icu_locale
```

- database creation:

```
CREATE DATABASE name LOCALE_PROVIDER = icu [ ICU_LOCALE [=]  
icu_locale ]
```

icu_locale is ICU locale ID, not PostgreSQL *collation* object name POSIX locale
needs to be set *as well*

pg15

no non-deterministic default collations

CREATE COLLATIONS: SYSTEM

Provided by external libraries with their rule definition sets:

libc

- `$ locale -a` to show all available locales
- `$ locale` to show current settings
- `$ locale -ck LC_ALL` to show it's definition
- `/etc/locale.gen` add/uncomment locale name and compile using `$ locale-gen` script
- use your package manager / vendor specific script `localectl`
- you can view the sources. Watch out for symlinks in generated locales

ICU

- APIs
 - `ucol_countAvailable()`
 - `ucol_getAvailable()`
 - `ucol_getDisplayName()`
- CLDR: Common Locale Data Repository
 - cldr.unicode.org, interactive browser
- LDML: Locale Data Markup Language (UTS #35)
 - github.com/unicode-org/cldr as Chart

CREATE COLLATIONS: DB

- first the rules need to be known to the OS
- during `initdb` all available collations are registered in `template0` catalog `pg_collation`, can be re-run later per DB: `pg_import_system_collations()`
 - `libc` `locale -a`, adds a less platform specific alias
 - `ICU` `uoloc_getAvailable()` and `uoloc_getDisplayName`, appends `-x-icu` to name
- `CREATE COLLATION` command

```
CREATE COLLATION [ IF NOT EXISTS ] name (  
    [ LOCALE = locale, ]  
    [ LC_COLLATE = lc_collate, ]  
    [ LC_CTYPE = lc_ctype, ]  
    [ PROVIDER = provider, ]  
    [ DETERMINISTIC = boolean, ]  
    [ VERSION = version ]  
)  
CREATE COLLATION [ IF NOT EXISTS ] name FROM existing_collation
```

- PROVIDER: `libc` / `icu`
- LOCALE: shortcut for `LC_COLLATE` and `LC_CTYPE`

```
SELECT * FROM pg_collation;
\dos+
```

```
[local] bussmann@~=# \dos+ de*
```

List of collations						
Schema	Name	Collate	Ctype	Provider	Deterministic?	Description
pg_catalog	de-AT-x-icu	de-AT	de-AT	icu	yes	German (Austria)
pg_catalog	de-BE-x-icu	de-BE	de-BE	icu	yes	German (Belgium)
pg_catalog	de-CH-x-icu	de-CH	de-CH	icu	yes	German (Switzerland)
pg_catalog	de-DE-x-icu	de-DE	de-DE	icu	yes	German (Germany)
pg_catalog	de-IT-x-icu	de-IT	de-IT	icu	yes	German (Italy)
pg_catalog	de-LI-x-icu	de-LI	de-LI	icu	yes	German (Liechtenstein)
pg_catalog	de-LU-x-icu	de-LU	de-LU	icu	yes	German (Luxembourg)
pg_catalog	de-x-icu	de	de	icu	yes	German
pg_catalog	de_AT	de_AT	de_AT	libc	yes	
pg_catalog	de_AT.UTF-8	de_AT.UTF-8	de_AT.UTF-8	libc	yes	
pg_catalog	de_CH	de_CH	de_CH	libc	yes	
pg_catalog	de_CH.UTF-8	de_CH.UTF-8	de_CH.UTF-8	libc	yes	
pg_catalog	de_DE	de_DE	de_DE	libc	yes	
pg_catalog	de_DE.UTF-8	de_DE.UTF-8	de_DE.UTF-8	libc	yes	
pg_catalog	default			default	yes	database's default collation

(15 rows)

LOCALE NAME SYNTAX

- **POSIX:**

```
language[_TERRITORY][.codeset][@modifier]
```

- `de_DE.ISO-8859-15@euro`
- `ca_ES.UTF-8@valencia`
- `de_CH.utf8`

- **BCP47:**

```
language[-Script][-REGION][-unicodeextension-x-privateuse]
```

- `de-u-co-phonebk-kn-true-ks-level2`
- `sr-Cyrl-XK` (Serbian, Cyrillic, Kosovo)
- `en`

- **Legacy ICU:**

```
[language[_Script][_REGION]][@key=value[;key=value]...]
```

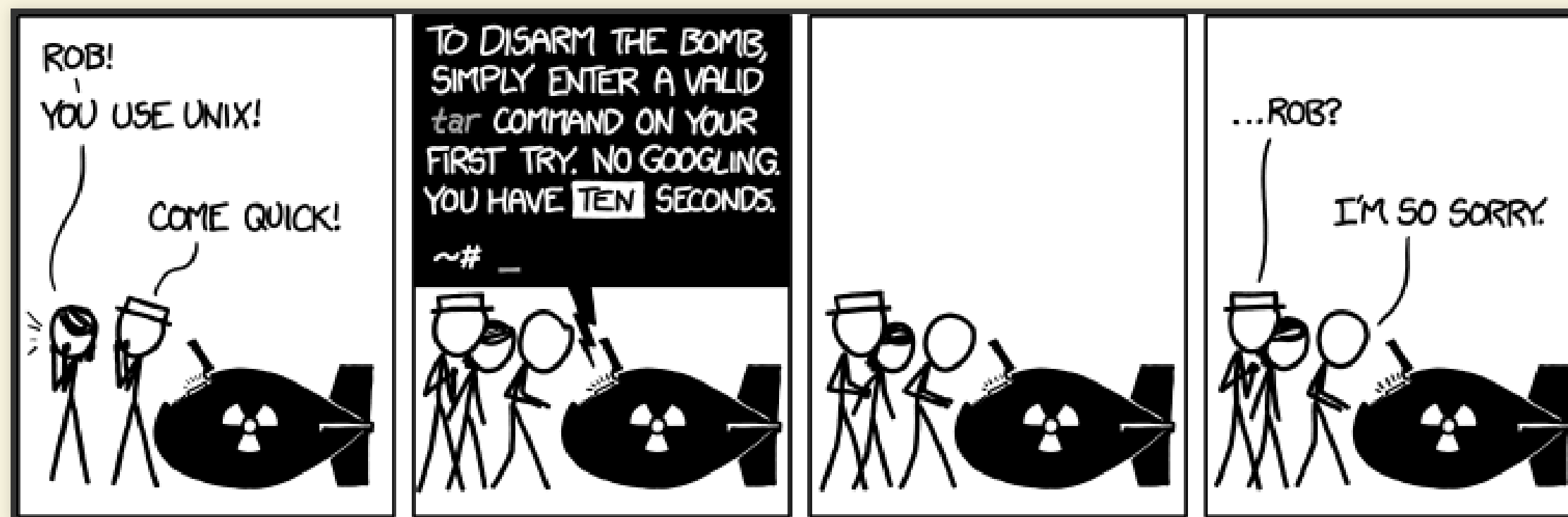
- `de_DE@collation=phonebook,colNumeric=yes,colStrength=secondary`
- `@collation=emoji` (using 'root' collation)
- `de@collation=phonebook`

BCP47 UNICODE EXTENSION

- Tailoring of collation behaviour from CLDR
- Syntax of defined in RFC6067:
[-u-key-type [-type] . . .] [-key-type [-type] . . .] . . .] keys unique
- keys and types defined in LDML / UTS#35, as XML (with alias names), interactive demo

key	type	description
co	standard, phonebk, search, trad, emoji, phonetic	Collation type. e.g. Traditional Spanish ordering, Phonebook ordering
ks	level1, level2, level3, level4, identic	Max collation strength: <i>Primary</i> : base letters, <i>Secondary</i> : accents, <i>Tertiary</i> : mainly case, <i>Quaternary</i> : used in some collations, <i>identical</i> : unicode codepoints (like ucs_basic)
kb	true, false	Sort second level (accents) backward: e.g. Canadian French
kc	true, false	Insert and use a strict case level between second and third level. Useful when ignoring accents (level1)
kf	upper, lower, false	Force to sort upper or lower case first
kk	true, false	Normalise to NFD before sorting
kn	true, false	Use natural sort for numbers. Useful for filenames, addresses.
kr	digit, space, grek, latn, ...	Order of character classes. Multiple types

...



SPECIAL COLLATIONS

not ICU dependent:

- "default": database default collation
- "C" = "POSIX": by encoded byte value
- "ucs_basic": for UTF8 by Unicode code points

ICU:

- "und-x-icu": 'root' collation (DUCET) with a reasonable language-agnostic sort order
UTS #10: UNICODE COLLATION ALGORITHM, as chart
- 'und-u-co-eor' European Ordering Rules
- 'und-u-co-emoji': 'root' collation with Emoji ordering (note: does conflict with languages), UTS #51: UNICODE EMOJI

USEFUL COLLATION DEFINITIONS

```
CREATE COLLATION "de-phonebook" (provider = icu, locale = 'de-u-co-phonebk');
CREATE COLLATION "de-natural" (provider = icu, locale = 'de-u-kn-true');
CREATE COLLATION "de-listing" (provider = icu, locale = 'de-u-co-phonebk-kn-true-ks-level2', deterministic = true);
CREATE COLLATION "und-emoji" (provider = icu, locale = 'und-u-co-emoji');
CREATE COLLATION "und-normalize" (provider = icu, locale = 'und-u-ko-normalize', deterministic = false);
CREATE COLLATION "und-nocase" (provider = icu, locale = 'und-u-ks-level2', deterministic = false);
CREATE COLLATION "und-noaccent" (provider = icu, locale = 'und-u-ks-level1-kc-true', deterministic = false);
CREATE COLLATION "und-noaccent-nocase" (provider = icu, locale = 'und-u-ks-level1', deterministic = false);
CREATE COLLATION "und-ignorepunctuation" (provider = icu, locale = 'und-u-ks-level3-ka-shifted', deterministic = true);
```

```
# SELECT * FROM
  regexp_split_to_table('a.b,aa,ab,a.d,a..b,a???c, ab')
ORDER BY x COLLATE "und-ignorepunctuation";
```

x
aa
ab
ab
a.b
a..b
a???c
a.d

```
# SELECT upper('i' COLLATE "tr-x-icu"),
  initcap('Bußmann' COLLATE "C");
```

upper	initcap
İ	BußMann

```
# SELECT * FROM
  regexp_split_to_table('file-1,file-10,file-2,file-9',
ORDER BY x COLLATE "de-natural";
```

x
file-1
file-2
file-9
file-10

WORKAROUNDS THAT MAY BE REPLACED

- `citext` extension
- `unaccent` extension
- Order / Compare / Index on function: `lower(x) / my_recursive_natural_sort(x)`
- ~~`ILIKE`~~
- Sort within application
- Many bugs due to unexpected, undetermined sort order
- More bugs due to different understanding what 'unique' means

THE BAD

Things to watch out for

PERFORMANCE

- Locale-aware comparisons are slower vs. locale C: `strcoll` is more expensive, needs NUL-terminated strings (requires `strcpy`) and possibly `strcmp` as tie-breaker.
- Using ICU may require encoding conversion.
- Several optimisations may not be usable:
 - *Abbreviated keys*, a powerful `btree` optimisation introduced with 9.5 had to be disabled in 9.5.2 due to bugs in `glibc`'s implementations of `strxfrm` in several locales. The API promise `strcoll(a,b) == strcmp(strxfrm(a), strxfrm(b))` didn't hold. It is still available in C (using only `strcmp` anyhow) or if compiled with `TRUST_STRXFRM`.
 - With ICU collations it is available again, often even faster and with a wider platform support.
 - *Non deterministic collations* (ICU only) are slower than deterministic ones, as they need to use the locale-aware comparison even if only equality needs to be tested.
 - But they should perform better than the functional / extension workarounds.

AVAILABILITY

- Locales need to be present in the OS
 - With ICU: `CREATE COLATION "superpower" (provider=icu, locale='invalid-like-hell');` does not throw an error. No check if collation definition is valid and in CLDR. Instead, ~~fuzzy~~ well-defined fallback till root collation.
 - Tip: Extension `icu_ext` provides `icu_collation_attributes()` to check how a Collation ID is interpreted by the library.
- `libc` collations need to match Encoding
- OS need not to lie about their collation support
 - there is no POSIX collation support for Unicode encoding in `*BSD`, incl. Darwin/macOS. (Illumos, Dragonfly and FreeBSD did some joint work in 2015)
- Non-Deterministic collations cannot be used as default collation

PATTERN MATCHING - PERFORMANCE

- locale aware index not useable for pattern matching, as collations are context-sensitive:

- in Czech alphabet: ...b, **c**, d, ... h, **ch**, i...

so abc < abcz < abch

```
# SELECT 'c' < 'd' COLLATE "cs-x-icu", 'ch' < 'd' COLLATE "cs-x-icu";
```

?column?	?column?
t	f

- an index using e.g. cs_CZ.UTF-8 is sorted accordingly and cannot be used to fulfil a condition like WHERE col LIKE 'abc%'.
- As a workaround either create an index with explicit COLLATE "C" or using the text_pattern_ops opclass. In that case location-unaware comparison operators are used: ~>~, ~>=~, ~<=~, ~<~
 - opclasses gin_trgm_ops / gist_trgm_ops for index-supported trigram pattern matching from the pg_trgm extension are not locale aware, too.

```
# \d cz_test
      Table "toolbox.cz_test"
```

Column	Type	Collation	Nullable	Default
t	text	cs_CZ		

```
Indexes:
    "cz_test_t_idx" btree (t)
    "cz_test_t_idx1" btree (t COLLATE "C")
    "cz_test_t_idx2" btree (t text_pattern_ops)
```

```
# EXPLAIN (COSTS OFF) SELECT * FROM cz_test WHERE t LIKE 'abc%';
```

QUERY PLAN
Index Only Scan using cz_test_t_idx2 on cz_test Index Cond: ((t ~>=~ 'abc'::text) AND (t ~<~ 'abd'::text)) Filter: (t ~~ 'abc%'::text)

```
# DROP INDEX cz_test_t_idx2;
# EXPLAIN (COSTS OFF) SELECT * FROM cz_test WHERE t LIKE 'abc%';
```

QUERY PLAN
Index Only Scan using cz_test_t_idx1 on cz_test Index Cond: ((t >= 'abc'::text) AND (t < 'abd'::text)) Filter: (t ~~ 'abc%'::text)

```
# DROP INDEX cz_test_t_idx1;
# EXPLAIN (COSTS OFF) SELECT * FROM cz_test WHERE t LIKE 'abc%';
```

QUERY PLAN
Seq Scan on cz_test Filter: (t ~~ 'abc%'::text)

PATTERN MATCHING - FEATURE

- pattern matching not possible for non-deterministic collations

```
# SELECT 'Bußmann' LIKE 'BUS%' COLLATE "und-nocase";  
ERROR: nondeterministic collations are not supported for LIKE
```

Why?

```
# SELECT upper('ß' COLLATE "de-DE-x-icu"), lower('SS' COLLATE "de-DE-x-icu");
```

upper	lower
SS	ss

(1 row)

- Same for ~
- As a workaround for case-insensitive matching: assign COLLATE "C" and use ILIKE or ~*
 - to speed-up, create an index with COLLATE "C" and opclass gin_trgm_ops / gist_trgm_ops to support

THE UGLY

avoid data corruption

COLLATION ORDER IS NOT FIXED.

- OS: Collation data does change. Collation providers get bugs fixes.
- DB: Order is persisted in indexes, behaviour is build upon.
- If these collide, bad things can happen - without even noticing.
 - **Index corruption:** Some queries may not find certain records anymore, Joins behave strangely. `SET enable_indexscan = OFF;` and data reappears
 - **Constraint violation:** Duplicated values in `UNIQUE` / PK column. `CHECK` -Constraints not respected.
 - **Partition routing:** rows are inserted or searched in the wrong partition
 - Unlikely PostgreSQL will throw an error about that.
- Different OS / collation provider versions: Same query, same data but different results

LIBC UPDATES

- Depending on the policy of the OS distribution. Likely during major updates, bugfixes may be included earlier
- **glibc 2.28** (released 2018-08-01) ~~was~~ is a particularly dangerous update.
 - Updates locale data according to *ISO/IEC 14651:2016*, which was synchronised with Unicode 9
 - Last big update in 2000/2001, since then only minor changes in single collations on a case-by-case basis
 - Changes lots of popular collations (even `en_US`) in an obvious way. Minimal test: `(echo "1-1"; echo "11") | LC_COLLATE=en_US.UTF-8 sort`
 - Known to be updated in Debian 10, Ubuntu 18.10, RHEL & CentOS 8, Fedora 29, ...
 - Further updates to be expected *ISO/IEC 14651:2019*

- **At risk** if `libc` / OS is updated and:
 - PostgreSQL `data` directory kept (same or only minor PostgreSQL update)
 - `data` directory updated using `pg_upgrade`
- **At risk** if running multiple servers with different versions and:
 - Using physical / streaming replication (standby affected)
 - Restoring physical backups made on a different environment (e.g. `pg_basebackup`)
- **No risk** if:
 - running in `C` locale only
 - restoring from logical backup (`pg_dump`)
 - using logical replication (receiver unaffected)

VERSIONING TO THE HELP

- *most* collation providers support versioning of collation data
- Version is recorded in `pg_collation.collversion` when collation is created
- Current version can be checked using `pg_collation_actual_version()`
- This check is done when the collation is first *used* after start, in case of mismatch a warning is emitted:

```
WARNING: collation "name" has version mismatch
```

- After **manually dealing with the issue**, issue

```
ALTER COLLATION name REFRESH VERSION;
```

```
ALTER DATABASE name REFRESH COLLATION VERSION;
```

- It helps if packager of PostgreSQL maintains a dedicated `libcicu`, too

MITIGATIONS AFTER COLLATION CHANGE

- `REINDEX` all indexes on `text`, `varchar`, `char`, and `citext` that are not using one of the collations `C`, `POSIX`, `ucs_basic` (deterministic)
 - not easily possible to decide if necessary, `amcheck` extension may help
- review partitioning keys for `PARTITION BY RANGE`. If affected, reroute tuples manually or run `pg_dump` with option `--load-via-partition-root`
- Do logical replication to new database / cluster

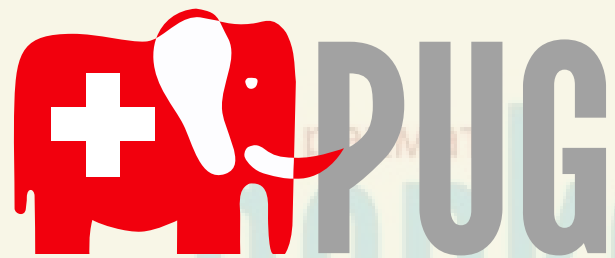
HISTORY / FUTURE

HISTORY OF LOCALE SUPPORT

- 6.1 (1997): initial cluster wide locale support (*Oleg Bartnunov, ...*)
- 8.1 (2005): initial ICU support, replacing the POSIX one by a patch for FreeBSD port (*Palle Girgensohn*)
- 8.3 (2008): FreeBSD patch updated to UTF8 to eliminate conversion to UTF16 (*Petr Jelinek, Palle Girgensohn*)
- 8.4 (2009): database-level `LC_COLLATION` and `LC_CTYPE` (*Heikki Linnakangas, Radek Strnad*)
- 9.1 (2011): collation support for columns, domains, and expressions, `COLLATE` clause, B-tree index support. (*Peter Eisentraut*)
- 9.6 (2016): FreeBSD patch updated with column and expression support (*Palle Girgensohn*)
- 10 (2017): Collation provider infrastructure, ICU collation support (different implementation from FreeBSD) (*Peter Eisentraut*)
- 12 (2019): non deterministic collations for ICU (*Peter Eisentraut*)
- 13 (2020): glibc & Windows Collation Version Support (*Thomas Munro*), Unicode normalizing functions (*Peter Eisentraut*)
- 14 (2021): BSD Collation Version support (*Thomas Munro*)
- 15 (2022): ICU Collations as default for cluster/db, per DB version support (*Peter Eisentraut*)

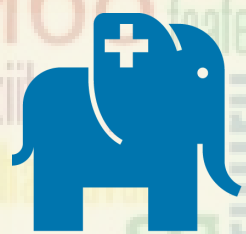
FUTURE

- Allow to load multiple version of the ICU library at runtime
- Louder error on collation mismatch
- Move version tracking from the collation object to individual database objects that use it.
Complex patch has been reverted from pg14 *Julien Rouhaud, Thomas Munro*
 - finer granularity
 - Tracking of use & version of default collation
- Improve handling of `pg_upgrade` regarding collation versions in catalog



Swiss PostgreSQL Users Group

www.swisspug.org



PG Day

Swiss PGDay 2023: tba.

www.pgday.ch



Tobias Bussmann

t.bussmann@gmx.net



GitHub

[tbussmann](https://github.com/tbussmann)



Twitter

[TobiasBussmann](https://twitter.com/TobiasBussmann)