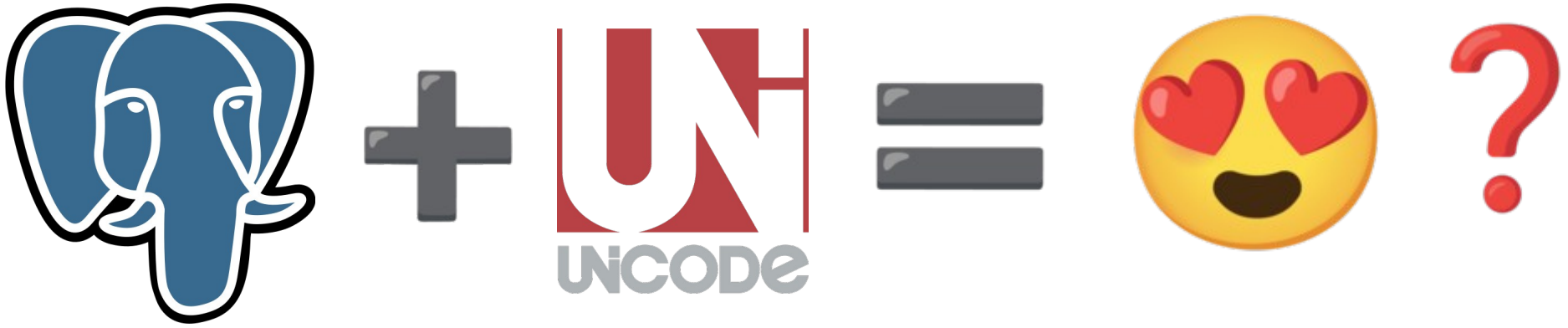
# PgDay France 2023

## PostgreSQL & Unicode



@DanielVerite

# A propos de moi

Daniel Vérité -- daniel@manitou-mail.org

Consultant indépendant, expert bases de données spécialisé PostgreSQL.

**Projets PostgreSQL:**

- Contributeur occasionnel (psql et ICU)
- Extension ICU: https://github.com/dverite/icu_ext
- Mail en base de données: https://manitou-mail.org

**Présence en ligne**

- Blog Postgres en français: https://blog-postgresql.verite.pro
- Forum https://forums.postgresql.fr
- StackOverflow / dba.stackexchange.com: https://stackoverflow.com/users/238814
- Twitter @DanielVerite

# La mission d'Unicode

Unicode est d'abord un **projet** démarré en 1988 avec une mission: établir un **jeu universel de caractères**

- incluant toutes les langues
- comprenant tous les symboles textuels
- utilisable par toutes les plateformes et langages
- disponible sur tous les périphériques

# L'organisation Unicode

## Consortium, non-profit corporation

https://www.unicode.org/consortium/consort.html

**Full Members (Voting)**


**Institutional Members (Voting)**

# Unicode induit une norme ISO

**ISO/IEC 10646:2020**, 2800 pages.
https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

Définit l'Universal Coded Character Set (UCS) et les encodages UTF-8, UTF-16, UTF-32.

# Unicode = un standard évolutif

- Unicode 1.0 publié en 1991
- Unicode 15.0 publié en 2022
- 149186 points de code au total dans 327 blocs
- 161 scripts (latin, grec, thai, hiragana...)

https://www.unicode.org/versions/stats/

# Unicode : riche et complexe

https://www.unicode.org/reports/

- 14 documents "Standard Annexes"
- 8 documents "Technical Standards"
- 7 documents "Technical Report"

# Intégration Unicode dans PostgreSQL

Le code de PostgreSQL gère l'encodage UTF-8 mais délègue le reste à des bibliothèques.

• jusqu'avant la version 10: la bibliothèque C (libc) uniquement

• depuis la version 10: libc et ICU (International Components for Unicode)

# Initialisation avec libc

```
$ initdb [--locale=nom]
  ou [--lc-ctype=nom --lc-collate=nom]
 -D /chemin/vers/pgdata
```

Les noms de locales (voir `locale -a`):

- langue_REGION.encodage (ex: **fr_CA.utf8**)
- langue: code ISO 639
- REGION: code ISO 3166

# Initialisation par défaut

Par défaut c'est l'environnement ($LANG) qui détermine la locale des bases **`template0`** et **`template1`**

```
$ echo $LANG
fr_FR.UTF-8

$ /usr/lib/postgresql/15/bin/initdb -D data
...
L'instance sera initialisée avec la locale « fr_FR.UTF-8 ».
L'encodage par défaut des bases de données a été configuré en
conséquence avec « UTF8 ».
La configuration de la recherche plein texte a été initialisée
à « french ».
...
```

# Les locales non linguistiques

"C" ou "POSIX"

*« Les localisations "C" et "POSIX" sont portables, leur partie LC_CTYPE correspond au jeu de caractères ASCII 7 bits.»*

- **LC_COLLATE = C** trie par octets (non linguistique)
- **LC_CTYPE = C** ne connaît que le jeu de caractères US-ASCII (127 premiers caractères Unicode)

PostgreSQL gère ces locales en interne et garantit le même comportement sur tous les systèmes d'exploitation

# Locale C.UTF-8

Depuis GNU libc 2.35 (février 2022) la locale **C.UTF-8** ou **C.utf8** est définitivement gérée

- permet de trier par octets comme "C"
- connaît **tous les caractères Unicode** contrairement à "C"

Elle existe aussi sur des libc d'autres systèmes Unix (FreeBSD par exemple).

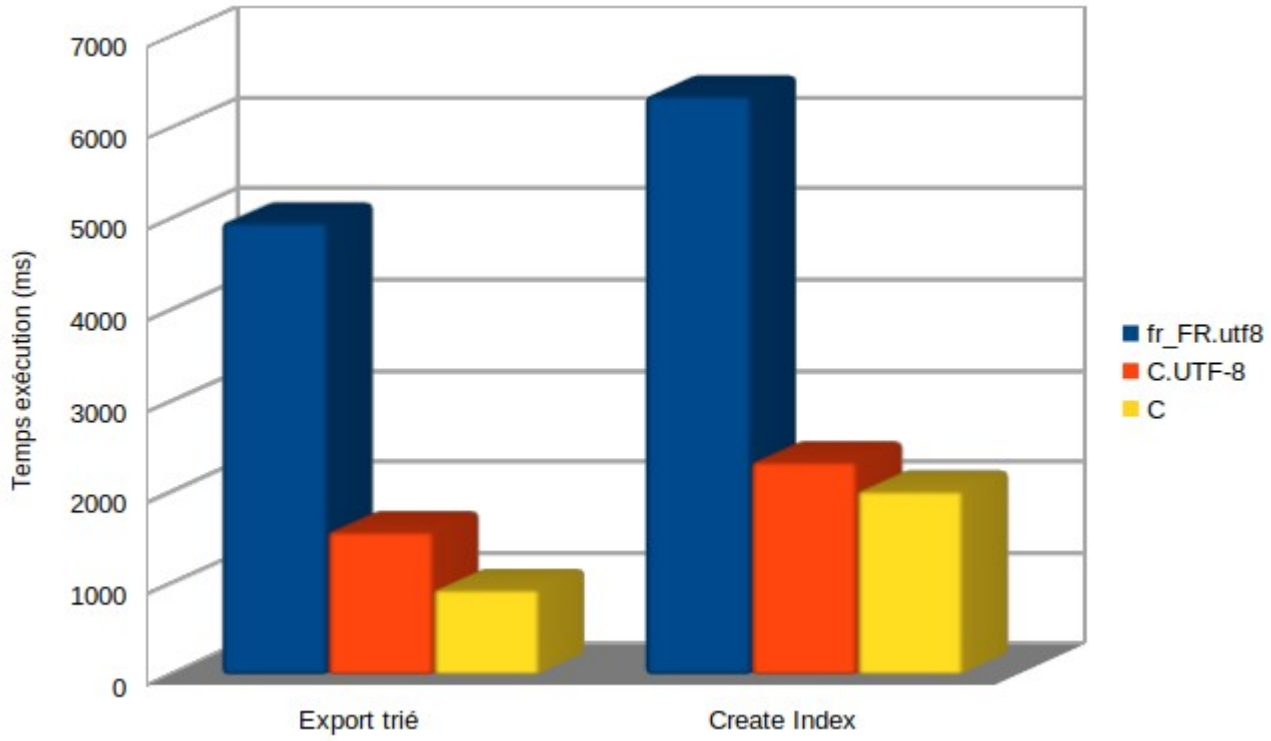
# Initialisation personnalisée

Il est intéressant de configurer le tri binaire non-linguistique (C) en conjonction avec la caractérisation des caractères Unicode.

```
$ /usr/lib/postgresql/15/bin/initdb -D data \
  --lc-collate=C --lc-ctype=C.UTF-8
...
L'instance sera initialisée avec cette configuration de locale :
  fournisseur:    libc
  LC_COLLATE:  C
  LC_CTYPE:    C.UTF-8
  LC_MESSAGES: fr_FR.UTF-8
  LC_MONETARY: fr_FR.UTF-8
  LC_NUMERIC:  fr_FR.UTF-8
  LC_TIME:     fr_FR.UTF-8
L'encodage par défaut des bases de données a été configuré en conséquence
avec « UTF8 ».
```

# Performance tri linguistique versus binaire

Exemple de tris sur des données de taille moyenne (2,1 millions de lignes, 34 Mo de données brutes).



# Influence langue/région

Exemples de différences entre **fr_FR.utf8** et **fr_CA.utf8**:

```
WITH x(mot) AS (VALUES
   ('16devel'), ('16~~devel'),
   ('1A'), ('1a'),
   ('1-2ans'), ('12 ans'),
   ('amené'),('amène'))
SELECT mot FROM x
ORDER BY mot ;
```

Exécutons cette requête sur Linux Ubuntu 22.04 (GNU libc 2.35)

# Influence langue/région

Locale "fr_FR.utf8"

```
 mot
--------
 12 ans
 1-2ans
 16devel
 16~~devel
 1a
 1A
 amené
 amène
```

Locale "fr_CA.utf8"

```
 mot
--------
 1-2ans
 12 ans
 16~~devel
 16devel
 1A
 1a
 amène
 amené
```

Locale "C.utf8"

```
 mot
--------
 1-2ans
 12 ans
 16devel
 16~~devel
 1A
 1a
 amené
 amène
```

# Tri linguistique, voulu ou subi ?

A propos du coût du tri linguistique versus tri binaire:

R.Haas blog, 2012, "The Perils of Collation-Aware Comparisons":

« *I suspect there are a lot of people out there who are paying it more or less accidentally and don't really care very much about the underlying sorting behavior*»

https://rhaas.blogspot.com/2012/03/perils-of-collation-aware-comparisons.html

# Classification des caractères

La déclaration **LC_CTYPE** influe sur:

- les expressions régulières
- la mise en majuscules/minuscules
- l'analyseur lexical de la recherche plein texte

En principe, il n'y a aucune variabilité «culturelle» (langue/région) des résultats, contrairement aux tris.

# Variabilité LC_CTYPE

D'un système à l'autre, la classification des caractères n'est pas 100% identique.

Est ce que *TAMIL DIGIT ONE* (U+B0E7) ou
*MATHEMATICAL SANS-SERIF DIGIT ONE* (U+1D7E3) sont des chiffres?

- Linux glibc 2.35, collation fr_FR.UTF-8

```
SELECT '௧' ~ '\d' ;
=> false
SELECT '𝟣' ~ '\d' ;
=> false
```

- FreeBSD 12.4, collation fr_FR.UTF-8

```
SELECT '௧' ~ '\d' ;
=> true
SELECT '𝟣' ~ '\d' ;
=> true
```

# Création d'une base

Une base peut remplacer les locales choisies à la création de l'instance.

```
CREATE DATABASE nom
  [LOCALE = ...]
  ou bien [LC_COLLATE =... LC_CTYPE = ...]
TEMPLATE = 'template0' ;
```

Si la locale n'est pas celle de la création de l'instance, il faut se référer à *template0* qui est la base modèle "100% d'origine".

# Création d'une collation

On peut utiliser d'autres locales que celle de la base en créant des collations à l'intérieur de la base.

```
CREATE COLLATION [schema.]nom
  [LOCALE = ...]
  ou [LC_COLLATE = ... LC_CTYPE = ...] ;
```

Généralement ce n'est pas utile car **initdb** importe les collations liées aux locales accessibles via libc dès l'initialisation de l'instance.

# Clause COLLATE 1/2

Toute déclaration de colonne de type texte peut être suivie d'une clause COLLATE

```
CREATE TABLE produit (
  code_produit TEXT COLLATE "C"
  ...
);
```

Si un index est créé sur cette colonne, il sera trié en **binaire** (par ordre des points de code).

# Clause COLLATE 2/2

Toute expression de type texte peut être suivie d'une clause COLLATE désignant un nom de collation:

```
SELECT * FROM produit
 ORDER BY code_produit COLLATE "default";

SELECT 'abc' < 'ABC' COLLATE "C";

=> false

SELECT upper('été' COLLATE "C");

=> éTé
```

# Premiers pas vers ICU

ICU a été introduit comme «fournisseur de collation» supplémentaire dans PostgreSQL 10 (2017)

Pourquoi offrir une alternative à libc?

# Bug des clefs abrégées

Incompatibilité entre *strxfrm()* et *strcoll()* sur la libc GNU (Linux) engendrant des corruptions d'index avec Postgres 9.5.0

https://wiki.postgresql.org/wiki/Abbreviated_keys_glibc_issue

Conclusion: désactivation de la fonctionnalité des « clefs abrégées »

# Bug de non-transitivité

Sur la libc Windows, la comparaison linguistique sur certains cas particuliers présente l'anomalite suivante:

- chaine1 < chaine2
- chaine2 < chaine3
- chaine1 > chaine3

Discussions sur pgsql-hackers:

*Windows UTF-8, non-ICU collation trouble*

https://postgr.es/m/20191206063401.GB1629883%40rfd.leadboat.com

*Inconsistent results with libc sorting on Windows*

https://postgr.es/m/1407a2c0-062b-4e4c-b728-438fdff5cb07@manitou-mail.org

# libc hétérogènes

Pas d'homogénéité entre les plateformes: une locale de même nom trie différemment d'un système à l'autre.

Exemple:

```
select * from (values ('"0102"'), ('0102')) as x(x)
 order by x collate "en_US.utf8";
```

FreeBSD 11:

```
    x
--------
 "0102"
 0102
```

Debian 9:

```
    x
--------
 0102
 "0102"
```

Debian 10:

```
    x
--------
 "0102"
 0102
```

# Mise à jour glibc v2.28

En 2018 glibc version 2.28 a introduit un changement dans les tris linguistiques affectant considérablement l'ordre des résultats.

- Debian 9 → 10
- RHEL/CentOS 7 → 8

Conséquence: une mise à jour du système sans recréation des index textuels entraîne une corruption de ces index.

https://wiki.postgresql.org/wiki/Locale_data_changes

# Figer les collations

Solution de montée de version proposée par Amazon pour Linux RHEL/CentOS: décorréler la partie collation du reste de GNU libc

- Présentation PGCon Ottawa 2023

  https://www.youtube.com/watch?v=0E6O-V8Jato

  https://github.com/awslabs/compat-collation-for-glibc

- Code

  https://github.com/awslabs/compat-collation-for-glibc

# Limitations fonctionnelles de libc

Dans libc, pas de paramétrage des comparaisons de chaînes de caractères:

- insensible à la **casse**
- insensible aux **accents**

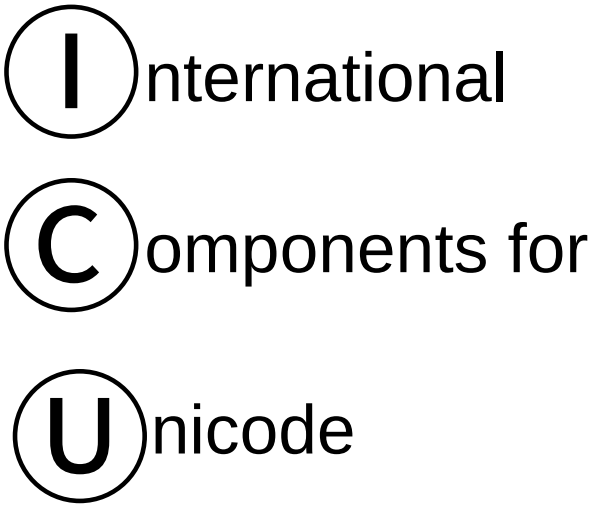
Solution de contournement via fonctions ou types spécifiques (extensions *unaccent* et *citext).*

# Absence de versionnage

Avec libc:

- Pas de versionnage systématique des locales
- Pas d'API pour récupérer un numéro de version par locale

# Qu'est-ce qu'apporte ICU ?




Bibliothèque C/C++/Java de référence pour Unicode

https://icu.unicode.org

# ICU: collations implicites

Au départ: **initdb** créé automatiquement environ 750 collations ICU déclinant 129 langues

- fr-x-icu (français, région indéfinie)
- fr-BE-x-icu (français parlé en Belgique)
- und-x-icu (langue non définie)
- unicode (Postgres version 16)

# Influence langue/région ICU

Exemples de différences entre
**fr-FR-x-icu** et **fr-CA-x-icu**:

```
WITH x(mot) AS (VALUES
   ('16devel'), ('16~~devel'),
   ('1A'), ('1a'),
   ('1-2ans'), ('12 ans'),
   ('amené'),('amène'))
SELECT mot FROM x
ORDER BY mot COLLATE "fr-FR-x-icu";
          -- suivi de "fr-CA-x-icu"
```

# Influence langue/région ICU

COLLATE "fr-FR-x-icu"

```
 mot
--------
 1-2ans
 12 ans
 16~~devel
 16devel
 1A
 1a
 amené
 amène
```

COLLATE "fr-CA-x-icu"

```
 mot
--------
 1-2ans
 12 ans
 16~~devel
 16devel
 1A
 1a
 amène
 amené
```

# Paramètres de collations

On peut influer sur le comportement des collations en jouant sur les paramètres linguistiques

```
CREATE COLLATION "fr-CA-bis" (
 PROVIDER = 'icu',
 LOCALE = 'fr-CA-u-kb-false'
);
```

La clé de paramétrage *kb-false* désactive la règle faisant que les accents sont comparés en partant de la fin de la chaîne.

# Paramètres des collations

COLLATE "fr-FR-x-icu"

| mot |
| --- |
| 1-2ans |
| 12 ans |
| 16~~devel |
| 16devel |
| 1A |
| 1a |
| amène |
| amené |

COLLATE "fr-CA-bis"

| mot |
| --- |
| 1-2ans |
| 12 ans |
| 16~~devel |
| 16devel |
| 1A |
| 1a |
| amène |
| amené |

# Paramétrage collations ICU

| Clef | Valeurs | Description |
|---|---|---|
| co | standard, emoji, phonebk | Type de tri |
| ka | noignore, shifted | Ignore certain caractères (ponctuation, espaces, ...) |
| kb | true, false | Tri des accents en partant de la droite |
| kc | true, false | Prendre en compte ou ignorer la casse |
| kf | upper, lower, false | Majuscules ou minuscules en premier |
| kn | true, false | Prise en compte des nombres |
| ks | level1, level2, level3, level4, identic | Niveau de comparaison (pour ignorer notamment casse et accents) |
| kv | space, punct, symbol, currency | Précise quels caractères ka-shifted doit ignorer |

Liste complète des paramètres prévus par le standard:
https://unicode.org/reports/tr35/tr35-collation.html#Setting_Options

# Versionnage des collations

Toutes les collations ICU ont un numéro de version généré automatiquement.

```
SELECT collversion FROM pg_collation
 WHERE collname = 'fr-CA-bis'
   AND collnamespace = 'public'::regnamespace;

 collversion
-------------
 153.112.40
```

# Avertissement de changement de version

Exemple de montée de version du système d'exploitation en gardant la même instance PostgreSQL

```
SELECT 'abc' < 'ABC' COLLATE "fr-x-icu" ;

ATTENTION:  le collationnement « fr-x-icu » a des versions différentes

DÉTAIL : Le collationnement dans la base de données a été créé en utilisant la version 153.14 mais le système d'exploitation fournit la version 153.112.

ASTUCE : Reconstruisez tous les objets affectés par ce collationnement, et lancez ALTER COLLATION pg_catalog."fr-x-icu" REFRESH VERSION, ou construisez PostgreSQL avec la bonne version de bibliothèque.
```

# Table système pg_collation

## Evolutions de la structure

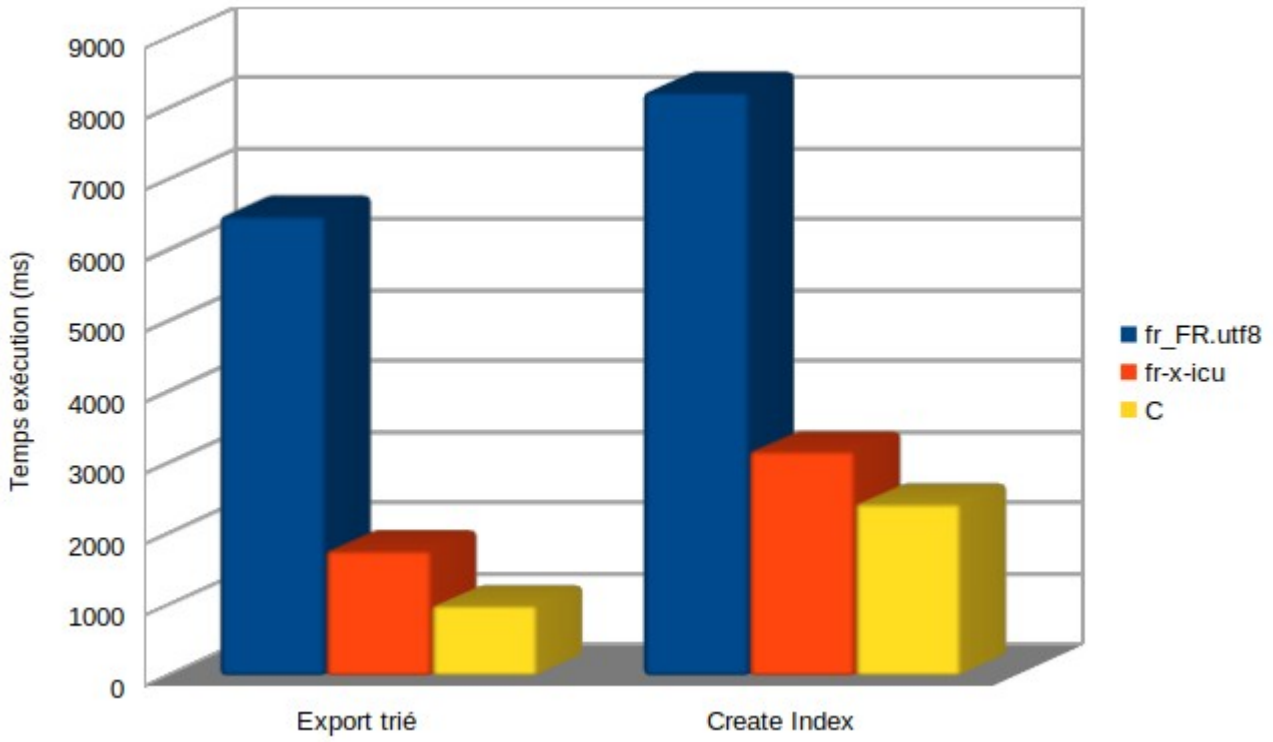
Table « pg_catalog.pg_collation »

| Colonne | Type | Collationnement | NULL-able |
|---|---|---|---|
| oid | oid | | not null |
| collname | name | | not null |
| collnamespace | oid | | not null |
| collowner | oid | | not null |
| collprovider | "char" | | not null |
| collisdeterministic* | boolean | | not null |
| collencoding | integer | | not null |
| collcollate | text | C | |
| collctype | text | C | |
| colliculocale* | text | C | |
| collversion | text | C | |

(*) *collisdeterministic* à partir de PostgreSQL 12
*colliculocale* à partir de PostgreSQL 15, avant utiliser *collcollate*

# Performances des tris

Exemple de tris sur des données de taille moyenne (2,1 millions de lignes, 34 Mo de données brutes).




- fr_FR.utf8: collation glibc 2.28 (Debian 10)
- fr-x-icu: ICU 73
- collation "C" (binaire): tri internalisé dans Postgres

# Collations non déterministes

Pour bénéficier complètement du paramétrage des collations, il faut désactiver le court-circuitage de Postgres qui pré-suppose que:

« Si deux chaînes ont une représentation binaire différentes, elles ne peuvent pas êtres égales ».

Les collations non déterministes introduites en **version 12**, permettent d'outrepasser cette règle.

# Collations insensibles à la ponctuation

```
CREATE COLLATION "no-punct" (
 PROVIDER = 'icu',
 LOCALE = 'und-u-ka-shifted',
 DETERMINISTIC = false
);

SELECT 'Pg-Day' = 'PgDay' COLLATE "no-punct";
=> true
```

# Collations insensibles à la casse

```
CREATE COLLATION "niveau2" (
 PROVIDER = 'icu',
 LOCALE = 'und-u-ks-level2',
 DETERMINISTIC = false
);

SELECT 'PgDay' = 'PGDAY' COLLATE "niveau2";
=> true
```

# Combinaison de paramètres

```
CREATE COLLATION "matcher" (
 PROVIDER = 'icu',
 LOCALE = 'und-u-ks-level1-kc-true-ka-shifted',
 DETERMINISTIC = false
);
```

- **ks-level1**: comparaison au niveau primaire (lettres de base)
- **kc-true**: prendre en compte la casse (majuscules/minuscules)
- **ka-shifted**: ignorer les caractères de ponctuation

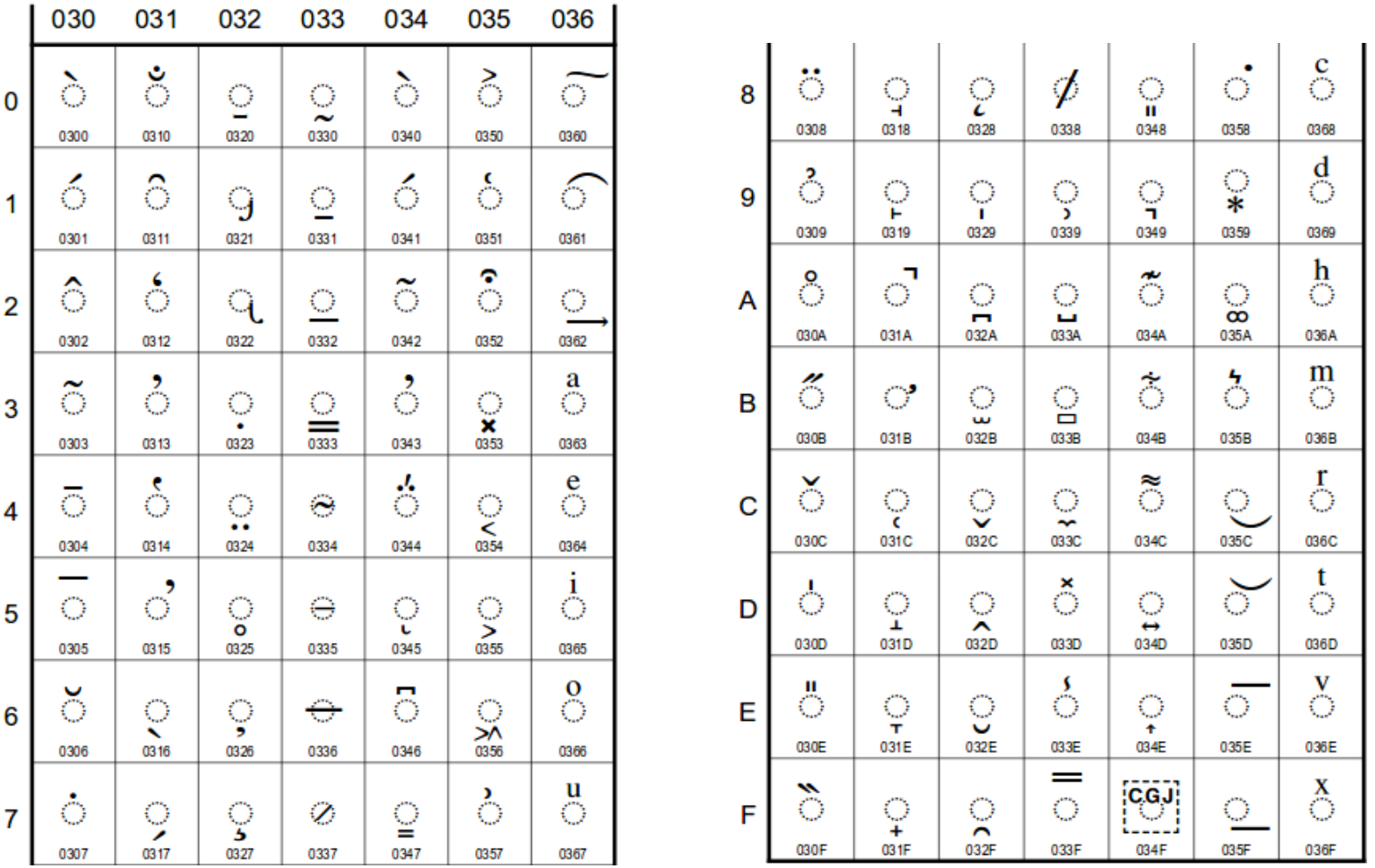
# Équivalence canonique

https://fr.wikipedia.org/wiki/Équivalence_Unicode

*«L'équivalence canonique est une forme d'équivalence qui préserve visuellement et fonctionnellement les caractères équivalents. Ils ont un codage binaire différents mais représentent un texte identique*.»

# Diacritiques combinants

U+0300 → U+036F https://www.unicode.org/charts/PDF/U0300.pdf



# Diacritiques combinants

```
CREATE COLLATION "nd" (
 PROVIDER = 'icu',
 LOCALE = 'fr',
 DETERMINISTIC = false
);

SELECT U&'a\0300' AS decomp,
  U&'a\0300' = 'à' COLLATE "default" egal_determ,
  U&'a\0300' = 'à' COLLATE "nd" egal_non_determ
;

 decomp | egal_determ | egal_non_determ
--------+-------------+-----------------
 à      | f           | t
```

# Normalisation Unicode

A partir de PostgreSQL 13, la normalisation est gérable via des fonctions

- **normalize**(texte, *FORM* ): transformation dans la forme indiquée
- texte IS [NOT] *FORM*

Forme principalement utilisée: NFC (forme composée).

# Limitations des collations non déterministes

- La clause LIKE n'est pas fonctionnelle.
- Les expressions régulières non plus.
- Elles ne peuvent pas être en collation par défaut d'une base.
- Les tests d'égalité sont nettement moins rapides que pour les collations déterministes.

# PostgreSQL 15 - Base ICU

On peut créer des bases dont la locale par défaut est gérée par ICU

```
CREATE DATABASE nom
  LOCALE_PROVIDER = 'icu'
  ICU_LOCALE = 'fr'
  TEMPLATE = 'template0' ;
```

# PostgreSQL 15 - instance ICU

On peut créer une instance dont la locale par défaut est gérée par ICU

```
$ /usr/lib/postgresql/15/bin/initdb -D data \
  --locale-provider=icu --icu-locale=fr-FR
...
L'instance sera initialisée avec cette configuration de locale :
  fournisseur:    icu
  locale ICU :  fr-FR
  LC_COLLATE:  fr_FR.UTF-8
  LC_CTYPE:    fr_FR.UTF-8
  LC_MESSAGES: fr_FR.UTF-8
  LC_MONETARY: fr_FR.UTF-8
  LC_NUMERIC:  fr_FR.UTF-8
  LC_TIME:     fr_FR.UTF-8
L'encodage par défaut des bases de données a été configuré en conséquence
avec « UTF8 ».
```

# Le futur

- ICU par défaut (à tester dans 16-beta2)?
- Meilleure intégration de C.UTF-8?
- Tri binaire (locale=C/POSIX) avec ICU en gestion de caractères?
- Collations non déterministes acceptées au niveau de la base?
- Multi-versions d'ICU pour lisser la réindexation?