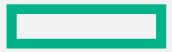


Hewlett Packard
Enterprise

ICU Locale の話

Noriyoshi Shinoda

September 9, 2022

SPEAKER
篠田典良（しのだのりよし）

- 所属
  - 日本ヒューレット・パッカード合同会社
- 現在の業務
  - PostgreSQL をはじめ Oracle Database, Microsoft SQL Server, Vertica 等 RDBMS 全般に関するシステムの設計、移行、チューニング、コンサルティング
  - Oracle ACE（2009 年 4 月～）
  - オープンソース製品に関する調査、検証
- PostgreSQL 15 に対して
  - ドキュメントの修正などのパッチ（8件）
- 関連する URL
  - 「PostgreSQL 虎の巻」シリーズ
    - http://h30507.www3.hp.com/t5/user/viewprofilepage/user-id/838802
  - Oracle ACE ってどんな人？
    - http://www.oracle.com/technetwork/jp/database/articles/vivadeveloper/index-1838335-ja.html

SPEAKER
篠田典良（しのだのりよし）

- PostgreSQL Unconference #15 （2020年9月30日）
  - 検知できない破壊の話
- PostgreSQL Unconference #20 （2021年2月2日）
  - プロセス障害の話
- PostgreSQL Unconference #29 （2021年12月21日）
  - 文字コードの話
- PostgreSQL Unconference #31 （2022年2月22日）
  - Babelfish の話
- PostgreSQL Unconference #35 （2022年9月9日）
  - ICU Locale の話
- スライドはこちら
  - https://www.slideshare.net/noriyoshishinoda
  - https://github.com/nori-shinoda/documents/blob/main/README.md

PostgreSQL 15 新機能
ロケール・プロバイダを選択可能に

- ロケール機能の提供ライブラリについて ICU と LIBC を選択できる
- 文字コードは UTF-8 で検証

Locale Provider のデフォルト設定

```
$ initdb -D data.libc --locale-provider=libc --locale=ja_JP.utf8 --encoding=utf8
$ initdb -D data.icu --locale-provider=icu --icu-locale=ja-JP --encoding=utf8
```

データベース単位の設定

```
postgres=# CREATE DATABASE dblibc1 LOCALE_PROVIDER=libc LC_COLLATE='ja_JP.utf8'
                LC_CTYPE='ja_JP.utf8' TEMPLATE=template0;
CREATE DATABASE
postgres=# CREATE DATABASE dbicu1 LOCALE_PROVIDER=icu ICU_LOCALE='ja-JP'
                TEMPLATE=template0;
CREATE DATABASE
```

PostgreSQL 15 新機能
ロケール指定パラメーター

– CREATE DATABASE 文で指定する項目とパラメーター

設定項目	LIBC パラメーター	ICU パラメーター	備考
ロケール・プロバイダ	LOCALE_PROVIDER		データベース作成時に決定
ロケール	LOCALE	ICU_LOCALE	データベース作成時に決定
ソート順	LC_COLLATE	-	データベース作成時に決定
文字の分類	LC_CTYPE	-	データベース作成時に決定
バージョン情報	COLLATION_VERSION		データベース作成時に決定・更新可
メッセージの言語	LC_MESSAGES		変更可能（初期値は initdbで決定）
通貨型の表示	LC_MONETARY		変更可能（初期値は initdbで決定）
数字の書式	LC_NUMERIC		変更可能（初期値は initdbで決定）
日時の書式	LC_TIME		変更可能（初期値は initdbで決定）

PostgreSQL 15 新機能
ICU ロケールのチェック

– ICU ロケール名や COLLATION バージョンの存在はチェックされていない？

```
$ initdb -D data.icu2 --locale-provider=icu --icu-locale=INVALIDNAME --encoding=utf8
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with this locale configuration:
  provider:    icu
  ICU locale:  INVALIDNAME
  ...
```

```
postgres=# CREATE DATABASE dbbad1 LOCALE_PROVIDER=icu ICU_LOCALE=INVALIDNAME
                COLLATION_VERSION='INVALIDVERSION' TEMPLATE=template0;
CREATE DATABASE
```

PostgreSQL 15 新機能
ICU ロケールのチェック

－COLLATION バージョン間違いは接続時に警告が出力される。

```
$ psql dbbad1 postgres
WARNING:  database "dbbad1" has a collation version mismatch
DETAIL:  The database was created using collation version INVALIDVERSION, but the operating system provides version 153.80.
HINT:  Rebuild all objects in this database that use the default collation and run ALTER DATABASE dbbad1 REFRESH COLLATION VERSION, or build PostgreSQL with the right library version.
psql (15beta4)
Type "help" for help.

dbbad1=# ALTER DATABASE dbbad1 REFRESH COLLATION VERSION;
NOTICE:  changing version from INVALIDVERSION to 153.80
ALTER DATABASE
```

ロケールの設定
money型の表示

－GUC 設定 lc_monetary に依存する(LIBC / ICU共通)

```
postgres=> SET lc_monetary = 'ja_JP';
SET
postgres=> SELECT 1000::money;
  money
---------
 ￥1,000
(1 row)
postgres=> SET lc_monetary='en_GB';
SET
postgres=> SELECT 1000::money;
   money
-----------
 £1,000.00
(1 row)
```

日本語ロケールの機能

ICU / LIBC 共通

- 全角アルファベットに対する upper / lower / initcap 関数の実行
- LIBC / ICU ロケール共通機能

```
dbicu1=> SELECT upper('半角：a'), upper('全角：ａ');
 upper    |  upper
----------+----------
 半角：A | 全角：Ａ
(1 row)

dbicu1=> SELECT lower('半角：A'), lower('全角：Ａ');
  lower   |  lower
----------+----------
 半角：a | 全角：ａ
(1 row)
```

日本語ロケールの機能
ICU / LIBC 共通

- 全角アルファベットに対する ILIKE 句の実行
- LIBC / ICU ロケール共通機能

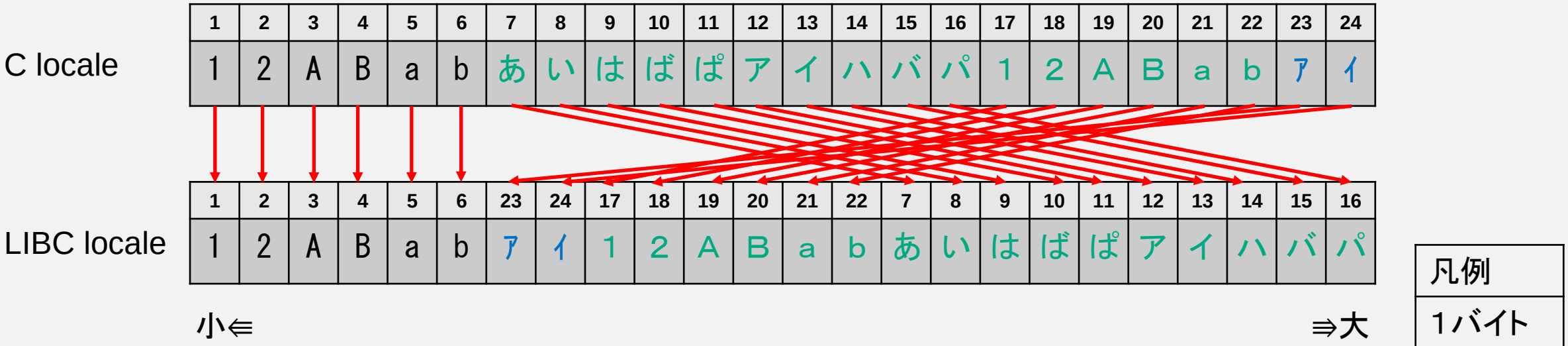
```
dbicu1=> SELECT * FROM data1 WHERE c2 ILIKE 'Ａ%';      -- 全角 Ａ
 c1 | c2
----+----
 19 | Ａ
 21 | ａ
(2 rows)

dbicu1=> SELECT * FROM data1 WHERE c2 ILIKE 'A%';       -- 半角 A
 c1 | c2
----+----
  3 | A
  5 | a
(2 rows)
```

日本語ロケールの機能

ソート順（LIBC ja_JP ロケール）

– C ロケール（= No Locale）と LIBC ja_JP ロケールの違い
– 1バイト文字 < 半角カナ < 全角（数字 < アルファベット < ひらがな < カタカナ）の順
– 同じカテゴリー内はほぼバイナリ順と同じ



日本語ロケールの機能

ソート順（ICU ja-JP ロケール）

- C ロケールと ICU ja-JP ロケールの違い
- 数字 < アルファベット < かな順
- 同じカテゴリー内は 半角 < 全角 < ひらがな < カタカナ < 半角カナ

凡例
1バイト
全角
半角カナ

そもそも
ロケール機能を推奨しない理由

- LIKE 句による前方一致でインデックスが使えない
  - C ロケールによる検索

```
postgres=> EXPLAIN SELECT * FROM data1 WHERE c2 LIKE '10000%';
                                 QUERY PLAN
-----------------------------------------------------------------------------
 Index Scan using idx1_data1 on data1  (cost=0.42..21.42 rows=100 width=10)
   Index Cond: (((c2)::text >= '10000'::text) AND ((c2)::text < '10001'::text))
   Filter: ((c2)::text ~~ '10000%'::text)
(3 rows)
```

  - LIBC ja_JP ロケールによる検索

```
dblibc1=> EXPLAIN SELECT * FROM data1 WHERE c2 LIKE '10000%';
                          QUERY PLAN
---------------------------------------------------------------
 Seq Scan on data1  (cost=0.00..17905.00 rows=100 width=10)
   Filter: ((c2)::text ~~ '10000%'::text)
(2 rows)
```

そもそも
ロケール機能を推奨しない理由

– インデックス作成時に {データ型} _pattern_ops 演算子クラスを指定することで回避可能

```
dblibc1=> CREATE INDEX idx1_data1 ON data1(c2 varchar_pattern_ops);
CREATE INDEX
dblibc1=> EXPLAIN SELECT * FROM data1 WHERE c2 LIKE '10000%';
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using idx1_data1 on data1  (cost=0.42..8.45 rows=100 width=10)
   Index Cond: (((c2)::text ~>=~ '10000'::text) AND ((c2)::text ~<~ '10001'::text))
   Filter: ((c2)::text ~~ '10000%'::text)
(3 rows)
```

– この場合、不等号「＜」「＞」による範囲検索でインデックス検索が行えなくなる

– 残念ながら ICU ロケールでも改善されていない

列単位でソート順を指定
CREATE TABLE 文

- 例） LIBC の場合 ja_JP COLLATION を指定

```
postgres=> CREATE TABLE data1 (c1 INT, c2 VARCHAR(10) COLLATE "ja_JP");
CREATE TABLE
```

- 例） ICU の場合 ja-JP-x-icu COLLATION を指定

```
postgres=> CREATE TABLE data1 (c1 INT, c2 VARCHAR(10) COLLATE "ja-JP-x-icu");
CREATE TABLE
```

- pg_collation カタログの collname 列に含まれる値を指定する
- ロケール指定時と同じ制約が発生

まとめ

LIBC ロケールと ICU ロケールの違い

- CREATE DATABASE 文では ICU ロケール名やバージョンのチェックが不十分
- ソート順以外は LIBC ロケールと ICU ロケールに違いは無さそう
- ロケール無しと比較すると LIBC と ICU で同じ制約がある
- ソート順を決めたい時は CREATE TABLE 文の COLLATE 句を指定で十分か？

THANK YOU

Mail: noriyoshi.shinoda@hpe.com
Twitter: @nori_shinoda