



# Collation Challenges

Sorting It Out

**Joe Conway**  
**conway@amazon.com**  
**mail@joeconway.com**

**AWS**  
**March 06, 2025**

# Agenda

- Problem with glibc collations
- How to fix it
- An alternate approach



## Setup on RHEL 7

```
initdb -D $PGDATA
pg_ctl -D $PGDATA start
psql postgres
psql (15.2)
Type "help" for help.
```

```
postgres=# SELECT pg_encoding_to_char(encoding) AS encoding,
               datlocprovider, datcollate, datctype, datcollversion,
               pg_database_collation_actual_version(oid) AS act_collversion
FROM pg_database WHERE datname = current_database();
```

```
-[ RECORD 1 ]----+-----
encoding      | UTF8
datlocprovider | c
datcollate    | en_US.UTF-8
datctype      | en_US.UTF-8
datcollversion | 2.17
act_collversion | 2.17
```



## Setup on RHEL 9

```
initdb -D $PGDATA
pg_ctl -D $PGDATA start
psql postgres
psql (15.2)
Type "help" for help.
```

```
postgres=# SELECT pg_encoding_to_char(encoding) AS encoding,
               datlocprovider, datcollate, datctype, datcollversion,
               pg_database_collation_actual_version(oid) AS act_collversion
FROM pg_database WHERE datname = current_database();
```

```
-[ RECORD 1 ]----+-----
encoding      | UTF8
datlocprovider | c
datcollate    | en_US.UTF-8
datctype      | en_US.UTF-8
datcollversion | 2.34
act_collversion | 2.34
```



## What's in a Sort

```
cat /etc/redhat-release && psql colltest << EOF
SELECT dat FROM (VALUES ('1-a'), ('1a'), ('1-aa')) v(dat) ORDER BY 1;
EOF
```

- RHEL 7

```
Red Hat Enterprise Linux Server release 7.9 (Maipo)
```

```
dat
-----
1a
1-a
1-aa
```

- RHEL 9

```
Red Hat Enterprise Linux release 9.0 (Plow)
```

```
dat
-----
1-a
1a
1-aa
```



## Create Table on RHEL 7

```
CREATE TABLE testcoll(f1 text primary key);
INSERT INTO testcoll (VALUES ('1-a'), ('1a'), ('1-aa'));
SELECT f1 FROM testcoll ORDER BY 1;
  f1
-----
 1a
 1-a
 1-aa
(3 rows)
```



## Upgrade OS to RHEL 9

```
SELECT f1 FROM testcoll ORDER BY 1;
   f1
-----
  1a
 1-a
1-aa
(3 rows)
```

```
INSERT INTO testcoll VALUES ('1-a');
INSERT 0 1
```

```
REINDEX TABLE testcoll;
2023-05-06 21:00:59.948 UTC [352755] ERROR:  could not create unique index "testcoll_pkey"
2023-05-06 21:00:59.948 UTC [352755] DETAIL:  Key (f1)=(1-a) is duplicated.
2023-05-06 21:00:59.948 UTC [352755] STATEMENT:  REINDEX TABLE testcoll;
ERROR:  could not create unique index "testcoll_pkey"
DETAIL:  Key (f1)=(1-a) is duplicated.
```



## Examine the Situation

```
SELECT pg_encoding_to_char(encoding) AS encoding,  
       datlocprovider, datcollate, datctype, datcollversion,  
       pg_database_collation_actual_version(oid) AS act_collversion  
FROM pg_database WHERE datname = current_database();  
-[ RECORD 1 ]---+-----  
encoding      | UTF8  
datlocprovider | c  
datcollate    | en_US.UTF-8  
datctype      | en_US.UTF-8  
datcollversion | 2.17  
act_collversion | 2.34
```





## One Way to Fix It

```
ALTER TABLE testcoll DROP CONSTRAINT testcoll_pkey;  
SELECT ctid, f1 FROM testcoll WHERE f1 = '1-a';
```

```
  ctid | f1  
-----+-----
```

```
(0,1) | 1-a
```

```
(0,4) | 1-a
```

```
(2 rows)
```

```
DELETE FROM testcoll WHERE ctid = '(0,4)';  
ALTER TABLE testcoll ADD PRIMARY KEY (f1);  
ALTER DATABASE colltest REFRESH COLLATION VERSION;  
NOTICE:  changing version from 2.17 to 2.34  
ALTER DATABASE
```

```
SELECT f1 FROM testcoll ORDER BY 1;  f1
```

```
-----
```

```
1-a
```

```
1a
```

```
1-aa
```



# Reexamine

```
SELECT pg_encoding_to_char(encoding) AS encoding,  
       datlocprovider, datcollate, datctype, datcollversion,  
       pg_database_collation_actual_version(oid) AS act_collversion  
FROM pg_database WHERE datname = current_database();  
-[ RECORD 1 ]---+-----  
encoding      | UTF8  
datlocprovider | c  
datcollate    | en_US.UTF-8  
datctype      | en_US.UTF-8  
datcollversion | 2.34  
act_collversion | 2.34
```



## Collation Torture Test - on RHEL 7

```
CREATE TABLE unsorted_table(strings text);
\copy unsorted_table from /home/ec2-user/formated-unicode.txt (format csv)
VACUUM FREEZE ANALYZE unsorted_table;
\timing
WITH t AS (SELECT strings FROM unsorted_table ORDER BY strings)
  SELECT md5(string_agg(t.strings,NULL)) FROM t;
           md5
-----
  7b2be833bc1893742f4b16d76d17e130
(1 row)
```

Time: 176505.256 ms (02:56.505)

See: <https://github.com/ardentperf/glibc-unicode-sorting>

And: <https://joeconway.com/presentations/formated-unicode.txt>



## Build Index - on RHEL 7

```
CREATE TABLE indexed_table(strings text);
INSERT INTO indexed_table SELECT strings FROM unsorted_table;
CREATE INDEX idx1 ON indexed_table(strings);
VACUUM FREEZE ANALYZE indexed_table;
\timing
WITH t AS (SELECT strings FROM indexed_table ORDER BY strings)
  SELECT md5(string_agg(t.strings,NULL)) FROM t;
          md5
```

-----  
[7b2be833bc1893742f4b16d76d17e130](#)

(1 row)

Time: 2988.474 ms (00:02.988)



## Check Index - on RHEL 7

```
CREATE EXTENSION amcheck;
```

```
SELECT bt_index_check('testcoll_pkey'::regclass, true);  
bt_index_check  
-----
```

```
(1 row)
```

```
Time: 1.616 ms
```

```
SELECT bt_index_check('idx1'::regclass, true);  
bt_index_check  
-----
```

```
(1 row)
```

```
Time: 46948.335 ms (00:46.948)
```



## Check Index - on RHEL 9

```
\timing  
WITH t AS (SELECT strings FROM unsorted_table ORDER BY strings)  
  SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
4ac498a5eb143e3991176ecf2f2132d4
```

```
(1 row)
```

```
Time: 3383540.995 ms (56:23.541)
```

```
WITH t AS (SELECT strings FROM indexed_table ORDER BY strings)  
  SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
7b2be833bc1893742f4b16d76d17e130
```

```
(1 row)
```

```
Time: 2733.324 ms (00:02.733)
```



## Check Index - on RHEL 9

```
\timing  
SELECT bt_index_check('testcoll_pkey'::regclass, true);  
ERROR: item order invariant violated for index "testcoll_pkey"  
DETAIL: Lower index tid=(1,1) (points to heap tid=(0,2)) higher index tid=(1,2)  
        (points to heap tid=(0,1)) page lsn=0/1903A88.  
Time: 2.398 ms
```

```
SELECT bt_index_check('idx1'::regclass, true);  
ERROR: item order invariant violated for index "idx1"  
DETAIL: Lower index tid=(411,9) (points to index tid=(1734,1))  
        higher index tid=(411,10) (points to index tid=(2021,1)) page lsn=0/EAE66D88.  
Time: 15.199 ms
```



## Fix Index - on RHEL 9

```
REINDEX TABLE testcoll;  
REINDEX  
colltest=# SELECT f1 FROM testcoll;
```

```
  f1  
-----  
 1-a  
 1a  
 1-aa  
(3 rows)
```

```
colltest=# SELECT bt_index_check('testcoll_pkey'::regclass, true);
```

```
-----  
  
(1 row)
```

bt\_





## FDW Issues - on RHEL 9

```
CREATE EXTENSION postgres_fdw;

CREATE SERVER foreign_server FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS (host 'jec-rh7', port '5432', dbname 'colltest', options '-c enable_seqscan=off');

CREATE USER MAPPING FOR "ec2-user" SERVER foreign_server
  OPTIONS (user 'ec2-user', password 'very secret pw');

CREATE FOREIGN TABLE f_testcoll (f1 text) SERVER foreign_server
  OPTIONS (schema_name 'public', table_name 'testcoll');
```



## FDW Issues - on RHEL 9

```
SELECT * FROM testcoll t JOIN f_testcoll f ON f.f1 = t.f1 ORDER BY 1;  
ERROR: mergejoin input data is out of order
```

```
EXPLAIN VERBOSE
```

```
SELECT * FROM testcoll t JOIN f_testcoll f ON f.f1 = t.f1 ORDER BY 1;  
QUERY PLAN
```

```
-----  
Merge Join (cost=100.13..174.84 rows=22 width=64)  
  Output: t.f1, f.f1  
  Inner Unique: true  
  Merge Cond: (f.f1 = t.f1)  
    -> Foreign Scan on public.f_testcoll f (cost=100.00..158.78 rows=1462 width=32)  
        Output: f.f1  
        Remote SQL: SELECT f1 FROM public.testcoll ORDER BY f1 ASC NULLS LAST  
    -> Index Only Scan using testcoll_pkey on public.testcoll t  
        (cost=0.13..12.18 rows=3 width=32)  
        Output: t.f1  
(9 rows)
```



## Partition Issues - on RHEL 7

```
CREATE TABLE testpart(f1 text not null) PARTITION BY RANGE (f1);
```

```
CREATE TABLE testpart_1 PARTITION OF testpart  
FOR VALUES FROM (MINVALUE) TO ('1-a');
```

```
CREATE TABLE testpart_2 PARTITION OF testpart  
FOR VALUES FROM ('1-a') TO (MAXVALUE);
```

```
INSERT INTO testpart VALUES ('1a');
```

```
SELECT * FROM testpart_1;
```

```
  f1  
----  
  1a  
(1 row)
```



## Partition Issues - on RHEL 9

```
INSERT INTO testpart VALUES ('1a');  
SELECT * FROM testpart_2;  
  f1  
----  
  1a  
(1 row)
```



## Why is it Important?

- Your collation probably provided by glibc in PG version 15 and earlier
- Sort order relies on collation
- Indexes persist sort order
- Constraints may depend on order
- PARTITION BY RANGE
- Some operations, e.g. mergejoin, depend on order



## Why is it Important?

- RHEL 7 EOL (glibc 2.17) → 30 June 2024
- Debian 10 EOL (glibc 2.28) → 30 June 2024
- Ubuntu 14.04 EOL (glibc 2.19) → April 2024



## Problems to Tackle

- Broken Indexes
  - Rebuild collation dependent indexes before any DML occurs
  - Otherwise, data loss may occur as cleanup may be needed
- Distributed Systems with Differing glibc versions
  - Replicas may have different glibc version
    - inconsistent ordering – depends on index used or not
    - failover implies broken indexes
  - Foreign Servers may have different glibc version
    - inconsistent ordering – broken mergejoins



## PG17 Builtin Provider

```
CREATE DATABASE builtincoll LOCALE_PROVIDER builtin
  BUILTIN_LOCALE 'C.UTF-8' TEMPLATE template0;
\c builtincoll
SELECT dat FROM (VALUES ('1-a'), ('1a'), ('1-aa')) AS v(dat) ORDER BY dat;
 dat
-----
 1-a
 1-aa
 1a
(3 rows)
```

```
SELECT dat FROM (VALUES ('1-a'), ('1a'), ('1-aa')) AS v(dat) ORDER BY dat COLLATE "en_US";
 dat
-----
 1-a
 1a
 1-aa
(3 rows)
```





## PG17 Builtin Provider

```
\timing  
WITH t AS (SELECT strings FROM unsorted_table ORDER BY strings)  
SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
7319ddb4b4a23ceabb70049cc6c61fe2
```

```
Time: 17859.714 ms (00:17.860)
```

```
\c glibc  
WITH t AS (SELECT strings FROM unsorted_table ORDER BY strings)  
SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
4ac498a5eb143e3991176ecf2f2132d4
```

```
Time: 3180930.438 ms (53:00.930)
```



## What is libcompatcollation?

- See: <https://github.com/awslabs/compat-collation-for-glibc>
- Method to build extracted glibc locale functionality into a library
- Pin to one glibc major or minor version
  - Provides immutable collation
- Standalone and portable to other Linux OS with same architecture
  - x86\_64 and aarch64 have been demonstrated successfully
- Use LD\_PRELOAD or build linked PostgreSQL



## How is it Created?

```
git clone git@github.com:awslabs/compat-collation-for-glibc.git
cd compat-collation-for-glibc/
git checkout 2.17-326.el7
./glibc-compatcollation.sh build
sudo rpm -ivh <path>/glibc-compatcollation217326-1.2-el7_9.x86_64.rpm
```



## Technical Details

- Applied on top of source RPM build
  - RPMs are built on upstream tarball + (many) patches
  - Preserve sorting semantics of very specific RPM package version
- Two distinct types of changes to the glibc RPM source
  - Changes to glibc source code
  - Changes to glibc package building code



## Changes to glibc Source Code

- Goal was to minimize the changes
- Types of changes in it fit into four categories
  - Fixing hardcoded assumptions about the paths for supporting-files
  - Allow non-locale glibc functionality to be sourced from a linked libc.so
  - Remove symbol versioning imposed by C code directives
  - Minor adjustments to standard libc functionality, e.g. `gnu_get_libc_version`



## Changes to glibc package building code

- `glibc.spec`
  - Provide libcompatcollation build instructions
  - Produce only libcompatcollation RPM
- Custom build support
  - `buildfiles.txt` - what glibc source files are included in the build
  - `libcompatcollation.map` - what symbols are exported
  - `build-compatcollation.sh` and `Makefile` - do the build



## build-compatcollation.sh and PRELOAD

- Edit `build-compatcollation.sh` to enable `LD_PRELOAD`
- Change `ENABLE_LD_PRELOAD=0` to `ENABLE_LD_PRELOAD=1`



## glibc Performance Regression

- Remember that horrible RHEL 9 sort timing?
- <https://sourceware.org/git/?p=glibc.git;a=commit;h=0742aef6>
- Prior to glibc 2.21, sorting lots of multibyte characters much faster
- Serendipity?





## It works!

- glibc collation version at 2.26-59
- Widely and successfully deployed:
  - Amazon Aurora PostgreSQL-Compatible Edition  
→ 16 (all), 15 (all), 14.6+, 13.9+, 12.13+, 11.18+
  - Amazon RDS for PostgreSQL  
→ 16 (all), 15 (all), 14.6+, 13.9+, 12.13+, 11.18+
  - See

<https://aws.amazon.com/blogs/database/manage-collation-changes-in-postgresql-on-amazon-aurora-and-amazon-rds/>



# Usage

- libcompatcollation in action



# Upgrade OS to RHEL 9 with libcompatcollation

```
cat /etc/redhat-release
Red Hat Enterprise Linux release 9.0 (Plow)

sudo rpm -ivh glibc-compatcollation217326-1.3-e17_9.x86_64.rpm
cd postgresql
./configure [...] LIBS="-lcompatcollation.2.17-326.e17_9"
make && make install
```

Also see: <https://joeconway.com/presentations/compat-collation.pg.15.patch>



## Test it Out

```
SELECT f1 FROM testcoll ORDER BY 1;
```

```
  f1
```

```
-----
```

```
 1a
```

```
1-a
```

```
1-aa
```

```
(3 rows)
```

```
INSERT INTO testcoll VALUES ('1-a');
```

```
ERROR: duplicate key value violates unique constraint "testcoll_pkey"
```

```
DETAIL: Key (f1)=(1-a) already exists.
```



## Examine the Situation

```
SELECT pg_encoding_to_char(encoding) AS encoding,  
       datlocprovider, datcollate, datatype, datcollversion,  
       pg_database_collation_actual_version(oid) AS act_collversion  
FROM pg_database WHERE datname = current_database();
```

| encoding | datlocprovider | datcollate  | datatype    | datcollversion | act_collversion |
|----------|----------------|-------------|-------------|----------------|-----------------|
| UTF8     | c              | en_US.UTF-8 | en_US.UTF-8 | 2.17           | 2.17-326.e17_9  |

(1 row)



## Check Index - RHEL 9

```
\timing  
WITH t AS (SELECT strings FROM unsorted_table ORDER BY strings)  
  SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
7b2be833bc1893742f4b16d76d17e130
```

```
(1 row)
```

```
Time: 177089.966 ms (02:57.090)
```

```
WITH t AS (SELECT strings FROM indexed_table ORDER BY strings)  
  SELECT md5(string_agg(t.strings,NULL)) FROM t;  
          md5
```

```
-----  
7b2be833bc1893742f4b16d76d17e130
```

```
(1 row)
```

```
Time: 2781.338 ms (00:02.781)
```



## Check Index - RHEL 9

```
\timing  
SELECT bt_index_check('testcoll_pkey'::regclass, true);  
  bt_index_check  
-----
```

(1 row)

Time: 2.052 ms

```
SELECT bt_index_check('idx1'::regclass, true);  
  bt_index_check  
-----
```

(1 row)

Time: 44118.175 ms (00:44.118)



## Check postgres Binary - RHEL 9

```
readelf -r ~/bin/postgres |\ngrep -E "(COMPATCOLL|GLIBC)" |\ntr -s " " |\ncut -d" " -f5 |\ntr "@" " " |\nsort -k2,2 -k1,1
```

```
bindtextdomain COMPATCOLL_1.0  
bind_textdomain_c[...] COMPATCOLL_1.0  
__ctype_b_loc COMPATCOLL_1.0  
__ctype_tolower_loc COMPATCOLL_1.0  
__ctype_toupper_loc COMPATCOLL_1.0  
...  
pwritev GLIBC_2.10  
memcpy GLIBC_2.14  
syncfs GLIBC_2.14  
clock_gettime GLIBC_2.17  
...
```





## Check All postgres Binaries - RHEL 9

```
#!/bin/bash
mappath="$HOME/<path-to-libcompatcollation.map>"
syms=$(sed -n '/COMPATCOLL_1.0/, $p' \
    $mappath/libcompatcollation.map |\
    tail -n +3|head -n -8|tr -d " ";")
objfiles=$(find . -name *.so)
objfiles="$objfiles $(find -type f -executable -exec file -i '' \; | \
    grep 'x-executable; charset=binary'|cut -d: -f1)"
for objfile in $objfiles
do
    echo "$objfile"
    for sym in $syms
    do
        found=$(objdump -T $objfile | grep -w $sym | grep LIBC)
        if [[ "$found" != "" ]]; then
            echo "    has symbol: $found"
        fi
    done
done
done
```



## Questions?

Thank You!  
mail@joeconway.com  
conway@amazon.com  
@josepheconway

