

GROUPBY 2021 | MAY 25-26

Free Online Training for Data Professionals.
By the Community, for the Community.

All You Wanted to Know about
Collations

Erland Sommarskog



GROUPBY CODE OF CONDUCT

The Quick Version

We are dedicated to a harassment-free experience for everyone, regardless of who you are and what makes you *you*. We recognize the right of any individual to attend and participate. Anyone. This is included but not limited to gender identity and expression, sexual orientation, disability, physical appearance, body size, race, religion, or any other classification, affiliation, or label.

We do not tolerate harassment in any form. For the duration of your engagement with GroupBy and its programs, you are expected to act appropriately and to adhere to this Code of Conduct. This includes conduct in-person and online, at the conference itself, as well as any non-conference programs that may include participants: including talks, workshops, parties, on social media, and other online forums. GroupBy participants violating these rules may be sanctioned or expelled without a refund (if that applies) at the discretion of the conference organizers.

You can review the full policy at: GroupBy.org/Code-of-Conduct





Erland Sommarskog

Independent consultant in Stockholm

SQL Server MVP since 2001

<http://www.sommarskog.se>

esquel@sommarskog.se

Slides and scripts on

<http://www.sommarskog.se/present>

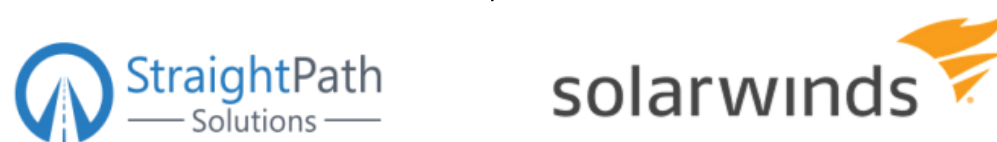


GROUPBY 2021 MAY 25-26

Core Sponsors



Media Sponsors



Virtual Group Sponsors



Agenda

- Collation basics.
- Unicode and code pages.
- The anatomy of a collation name.
- UTF-8 collations.
- Collations and metadata.
- Collation conflicts.
- Some useful tricks with collations.
- Collations and performance.

What Is a Collation?

A collation is a set of rules for how to handle string data depending on human language. This includes:

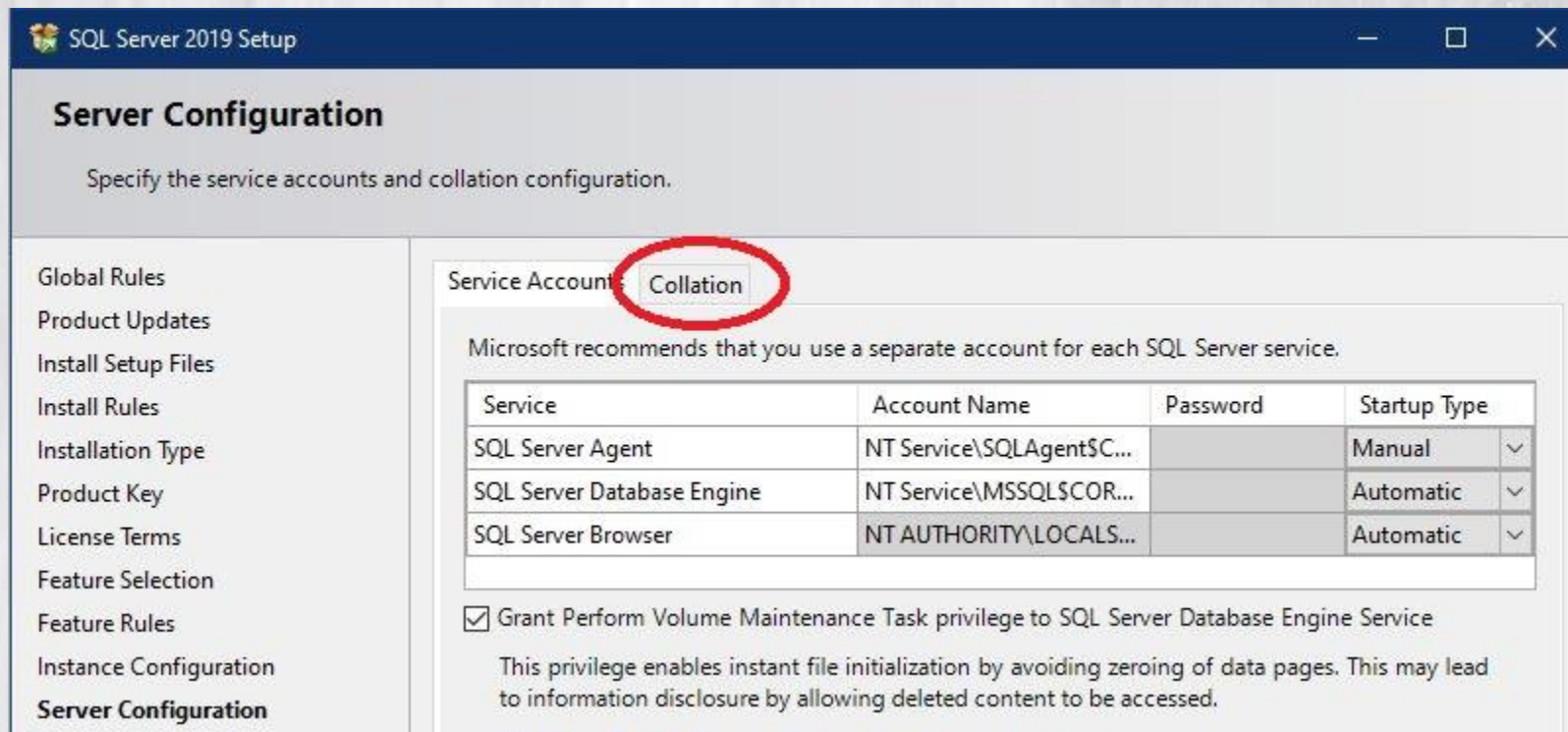
- Comparison. Is 'I' = 'i'? 'V' = 'W'? 'Ö' > 'Z'? '+' > '-'?
- Grouping and sorting.
- Result of upper/lower.
- LIKE ranges, for instance `col LIKE '[A-Z]%'`.
- Which code points in Unicode that are defined.
- The code page for varchar.
 - Not all collations support varchar!

Four Levels to Set Collations

- **Server collation** – set when you install SQL Server.
 - Collation for the system databases.
 - Default collation for string columns in temp tables.
 - The default for:
- **Database collation** – may override server collation.
 - Defines the collation for string variables and literals.
 - Sets the default for collation in user tables and table variables.
- **Column collation** – may override database setting.
- **Expression level** – cast collation on the fly.

Collations and Setup

- In Setup wizard, Collation is tucked away on a secondary tab:



Collations and Setup

- Always check what is on that secondary tab – it may not be what you expect.
- The default in the Setup Wizard is taken from the Windows system locale – which may differ from your personal regional settings.
- Changing a collation after the fact is painful!

Getting Collation Information

Server collation:

```
SELECT serverproperty('Collation')
```

Database collation:

```
SELECT databasepropertyex('MyDB', 'Collation')
```

Or look in sys.databases.

Column collation:

```
EXEC sp_help tablename
```

Or look in sys.columns.

The Unicode Character Set

- Windows is based on *Unicode*, a character set which supports all human languages.
- 1.1 million possible code points (21 bits). 143 000 are defined.
- New versions of Unicode define more code points.
 - A collation is based on a certain Unicode version.
- All living languages can expressed with the “base plane” – code points 0 to 65535 (16 bits).
- Code points 32-126 = ASCII. 160-255 = ISO Latin-1.

Unicode Encodings

- Sample code points: A = U+0041, Ł = U+0141, α = U+03B1, 中 = U+4E2D,  = U+1F0A1.
- Unicode can be encoded in several ways.
 - UTF-32, four bytes per code point, rarely used.
 - UTF-8, one to four bytes per code point.
 - UTF-16, two bytes per code point for the base plane, four bytes per code point for the rest.
- Windows is based on UTF-16.
- In SQL Server, nvarchar is UTF-16.

Legacy Code Pages

- Windows supports legacy charsets, called *code pages*.
- They all(?) have ASCII in the range 32-126 and use the range 127-255 to support one or more languages.
- Some examples:
 - CP850 (West Eur, proprietary from MS-DOS days).
 - CP932 (Japanese).
 - CP1250 (East Eur, ISO-8859-2, a.k.a. Latin-2).
 - CP1252 (West Eur, ISO-8859-1, a.k.a. Latin-1).
 - CP65001 (UTF8).
- These code pages are the foundation for varchar.
 - Varchar is not always 8-bit!

All the Collations

- In SQL 2019, there are 5508 (5681) collations.
- 5431 (5603) Windows collations.
 - Based on Windows system locales.
 - All operations are carried out in UTF-16 also for varchar.
- 77/78 SQL Collations, legacy from SQL 7 and earlier.
 - For varchar, based on a specific code page with its own library.
 - For nvarchar they are like a Windows collation.
 - While legacy, the most commonly used collation is most likely SQL_Latin1_General_CP1_CI_AS.

The Anatomy of a Collation Name

- Polish_100_CI_AI_KS_WS_SC_UTF8 – Means what?
- Polish – Collation family.
- 100 – Version number.
- CI/AI – Case and accent (in)sensitivity.
- KS/WS/VSS – Japanese/East Asian.
- SC – Surrogate compatible.
- BIN and BIN2 – Binary.
- UTF8 support.

Collation Families

- **Example of collation families:**
 - Polish, Assamese, Traditional_Spanish, Modern_Spanish, Finnish_Swedish, Latin1_General, Cyrillic_General.

A collation family determines:

- **Basic sorting and comparison rules.**
 - Further refined by CI/CS, AI/AS etc.
 - Possibly also by the version number.
- **Rules for lower/upper.**
- **The code page for varchar. (Not UTF8 collations.)**

The Version Number

- Can be none (=80), 90, 100 or 140.
 - 80: The original collations in SQL 2000.
 - 90: Support for a few more languages.
 - 100: New versions of the original and support for more languages.
 - 140: Only Japanese collations.
- The version number also specifies a version of Unicode – and thus which code points that are defined.

Case and Accent Sensitivity

- CI – Case insensitive. 'insert' = 'INSERT'.
- CS – Case sensitive. 'insert' <> 'INSERT'.
- AI – Accent insensitive. 'resume' = 'résumé'.
- AS – Accent sensitive. 'resume' <> 'résumé'.
- In a CS collation, case only has a secondary weight when sorting.
- Case sensitivity – Watch out for Turkish!
- Accent insensitive comes with surprises.

Japanese Matters – KS/WS/VSS

- **KS:** Japanese has two syllable scripts, Hiragana and Katakana.
 - Is なかめぐろ = ナカメグロ? (“Naka-Meguro”)
 - Yes, if no KS. No if KS (Kana-sensitive).
- **WS:** To match East-Asian and Latins characters in size, there are fullwidth and halfwidth forms.
 - Is Paris = P a r i s?
 - Is ナカメグロ = ナカメグロ?
 - Yes, if no WS. No if WS (Width-sensitive.)
- **VSS:** Variation-selector sensitive. I pass. :-)
 - Only in Japanese collations with version number = 140.

Binary Collations

- Binary collations sort and compare by code point.
- They are case- accent-, kana- and everything else-sensitive.
- Not always user-friendly – but they are fast.
- There are two of them `_BIN` and `_BIN2`.
- `BIN2` are “normal”, sort all by code point.
- `BIN` are legacy. Swap the first byte only, and sort remaining by raw binary.

[06_binarycollations.sql](#)

SC – Supplementary Characters or Surrogate Compatible

- In UTF-16, characters beyond the base plane are encoded as so-called surrogate pairs.
 - High word is in the range U+D800 to U+DBBB.
 - Low word is in the range U+DC00 to U+DFFF.
- The original version-80, 90 and 100 collations only support the Unicode base plane, 0-65535.
- The `_SC` collations support UTF-16 in full.
- All version 90 and later have an `_SC` version.
 - Version-140 collations are always surrogate compatible.
 - Binary collations are not surrogate aware.

UTF-8 Collations

- Introduced in SQL 2019. Suffix is `_UTF8`.
- All SC collations have a UTF-8 version.
 - Thus, there are no version-80 collations for UTF8.
- In a UTF8 collation the code page for varchar is always 65001 = UTF-8.
- UTF-8 collations do not support the text and ntext data types. (True for all SC collations.)
- There is one binary UTF-8 collation, `Latin1_General_100_BIN2_UTF8`.

Saving Space with UTF-8?

- The range 0-127 takes up one byte.
 - ASCII. Up to 50 % space saving for English.
- The range 128-2047 is encoded with two bytes.
 - Latin, Cyrillic, Greek, Arabic, and a few more scripts.
 - Still 5-10 % space saving for non-Latin scripts.
- The range 2048-65535 need three bytes.
 - Languages of India, Chinese, Japanese, Thai etc.
 - Up to 50 % more space.
- Beyond the base plane, four bytes.
 - No difference to UTF-16.

UTF-8 and String Length

- `varchar(30)` means 30 *bytes*.
- To permit for 30 characters, you may need `varchar(60)` – and still take your chances that you will not need to store Chinese or emojis.
- Then again, a 10-byte value in a `varchar(60)` does not take up more space than in a `varchar(30)`.

UTF-8 or UTF-16 (nvarchar)?

- Personally, for a new application I would still go with nvarchar for international support.
 - More predictable.
 - Better performance.
 - Space can be saved with row compression as well.
 - But it certainly is a matter of preference.
- For an existing app with varchar that faces international needs, moving to a UTF-8 collation can be less painful than switching to nvarchar.
 - But you may still need a lot of work.

Collations and Metadata

- Due to legacy, collations also control metadata.
- The server collation controls:
 - Names of server-level objects (databases, logins etc).
 - Names of temp tables and column names in temp tables and table variables.
 - *Names* of variables. (Values follow the database collation.)
- The database collation controls:
 - Names of database-level objects (tables, columns, users etc).
 - Service-Broker objects have a binary collation.

Tips for International Software

- Develop with a CS collation, both on server-level and database-level.
 - Or forget about the Turkish market.
- Corollary: Use only lowercase for identifiers.
- Avoid very short names like #v or @w to avoid accent-insensitive surprises.

Collation Conflicts

- When columns with different collations meet you get a conflict.

[11_collation_conflict.sql](#)

- Resolve with COLLATE:

```
JOIN a ON a.col COLLATE Latin1_General_CI_AS = b.col
```

- This casts the collation for *both* columns.
- Beware that casting the collation kills any index.
- Best Practice: Use `COLLATE database_default` in temp tables.
- Columns always win over variables/constants.

Useful Tricks with collations

- Removing “accents”.
- Finding words that starts with uppercase.
- Removing invisible characters.
- Casting to a non-Latin collation and then to varchar will replace accented characters with fallbacks.
- LIKE ranges depends on the collation, so you often need a binary collation.
- Some control characters are undefined in Windows collations – cast to a binary collation to resolve.

[12 useful tricks.sql](#)

Collations and Performance

[13_performance.sql](#)

- Lookup with nvarchar value on an indexed varchar column.
 - A programming mistake, but can easily happen.
- Windows collation: 2-5 times slower, but index still sought.
 - Possible because varchar is an ordered subset of nvarchar.
- SQL collation: index is scanned – performance disaster.
 - Because rules for varchar are different from nvarchar.

Performance with LIKE '%abc%'

- For varchar, SQL collation is 7-10 times faster than a Windows collation.
 - Windows collations and nvarchar are expensive, because the complex Unicode rules are applied to every character.
- Binary Windows collations are even faster – but are case-, accent- etc insensitive.
- Tip: Lower/upper() can make a binary search case-insensitive and good enough – and still as fast as an SQL collation.

Testing Performance

- I tested all 5508 collations.
- Test ran for nine days, testing five different operations, for both varchar and nvarchar.
- Platform was SQL 2019.
 - I also ran similar tests in 2008 on SQL 2008.
- Some selected data follows.
- Note: The effect on an real-world workload is likely be far less than the numbers indicate.

Data from Performance Tests

- CI_AI about 3-6 % faster than CS_AS.
- 100-collations 5-30 % slower than 80-collations.
 - Because character tables are larger.
- For Windows collations, varchar is 0-90 % slower than nvarchar.
 - Because of conversion to UTF-16 and back.
- For SQL collations, varchar is faster, if not always as extreme as with LIKE.
- Some languages are slower than others.

Performance of UTF8

- Comparing UTF-8 with varchar to nvarchar in version-80 collation:

- LIKE '%abc%': 12 % slower.
- Point lookup: 22 % slower.
- GROUP BY (hash match): 27 % slower.
- Loop join: 45 % slower.
- ORDER BY: 110 % slower.

This does not account for the space reduction with UTF-8. Take this with a big grain of salt.



GroupBy

Thank you! #organizers, #Moderators, #Attendees



- Nakshatra

The Final Slide

Erland Sommarskog
esquel@sommarskog.se

Slides and scripts on
<http://www.sommarskog.se/present>

Clean-up script: [14_cleanup.sql](#)