If you are going to change the Collation of a Database, then there is definitely stuff you should know about so that you can plan accordingly:

* Regarding data-loss potential:
  + NVARCHAR fields are all Unicode, which is a single character set, so there can't be any data loss for these fields (this also covers XML fields which are also stored as UTF-16 Little Endian). Meta-data fields that store the object / column / index / etc names are all NVARCHAR so no need to worry about those.
  + VARCHAR fields having different Collations but the same Code Page between the differing Collations will not be a problem since the Code Page is the character set.
  + VARCHAR fields having different Collations and moving to a different Code Page (when changing Collations) *can* have data loss if any of the characters being used are not represented in the new Code Page. HOWEVER, this is only an issue when physically changing the Collation of a particular field (described below) and would not happen upon changing the default Collation of a database.
* Local variables and string literals get their Collation from the Database default. Changing the database default will change the Collation used for both local variables and string literals. But changing the Database's default Collation does not change the Collation used for existing string columns in the tables in that Database. This generally should not cause any problems when comparing or concatenating a column with a literal and/or variable since the literals and variables will take on the Collation of the column due to Collation Precedence. The only potential problem would be Code Page conversions that might occur for characters of values between 128 - 255 that are not available in the Code Page used by the Collation of the column.
* If you are expecting a predicate / comparison / sort / concatenation / etc for a column to behave differently upon changing the Database's default Collation, then you will need to explicitly change that column's Collation using the following command:
* ALTER TABLE [{table\_name}]
* ALTER COLUMN [{column\_name}]
* {same\_datatype}
* {same\_NULL\_or\_NOT NULL\_setting}

COLLATE {name\_of\_Database\_default\_Collation};

Be sure to specify the *exact same* datatype and NULL / NOT NULL setting that are currently being used, else they can revert to the default if not already being the default value. After that, if there are any indexes on any of the string columns that just had their Collation changed, then you need to rebuild those indexes.

* Changing the Database's default Collation will change the Collation of certain database-specific meta-data, such as the name field in both sys.objects, sys.columns, sys.indexes, etc. Filtering these system Views against local variables or string literals won't be a problem since the Collation will be changing on both sides. But, if you JOIN any of the local system Views to temporary tables on string fields, and the Database-level Collation between the local database and tempdb doesn't match, then you will get the "Collation mismatch" error. This is discussed below along with the remedy.
* One difference between these two Collations is in how they sort certain characters for VARCHAR data (this does not affect NVARCHAR data). The non-EBCDIC SQL\_ Collations use what is called "String Sort" for VARCHAR data, while all other Collations, and even NVARCHAR data for the non-EBCDIC SQL\_ Collations, use what is called "Word Sort". The difference is that in "Word Sort", the dash - and apostrophe ' (and maybe a few other characters?) are given a very low weight and are essentially ignored unless there are no other differences in the strings. To see this behavior in action, run the following:
* DECLARE @Test TABLE (Col1 VARCHAR(10) NOT NULL);
* INSERT INTO @Test VALUES ('aa');
* INSERT INTO @Test VALUES ('ac');
* INSERT INTO @Test VALUES ('ah');
* INSERT INTO @Test VALUES ('am');
* INSERT INTO @Test VALUES ('aka');
* INSERT INTO @Test VALUES ('akc');
* INSERT INTO @Test VALUES ('ar');
* INSERT INTO @Test VALUES ('a-f');
* INSERT INTO @Test VALUES ('a\_e');
* INSERT INTO @Test VALUES ('a''kb');
* SELECT \* FROM @Test ORDER BY [Col1] COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS;
* -- "String Sort" puts all punctuation ahead of letters
* SELECT \* FROM @Test ORDER BY [Col1] COLLATE Latin1\_General\_100\_CI\_AS;

-- "Word Sort" mostly ignores dash and apostrophe

Returns:

String Sort

-----------

a'kb

a-f

a\_e

aa

ac

ah

aka

akc

am

ar

and:

Word Sort

---------

a\_e

aa

ac

a-f

ah

aka

a'kb

akc

am

ar

While you will "lose" the "String Sort" behavior, I'm not sure that I would call that a "feature". It is a behavior that has been deemed undesirable (as evidenced by the fact that it wasn't brought forward into any of the Windows collations). However, it *is* a definite difference of behavior between the two collations (again, just for non-EBCDIC VARCHAR data), and you might have code and/or customer expectations based upon the "String Sort" behavior. **This requires testing your code and possibly researching to see if this change in behavior might have any negative impact on users.**

* Another difference between SQL\_Latin1\_General\_CP1\_CI\_AS and Latin1\_General\_100\_CI\_AS is the ability to do [Expansions](http://www.unicode.org/reports/tr10/" \l "Expansions) on VARCHAR data (NVARCHAR data can already do these for most SQL\_ Collations), such as handling æ as if it were ae:
* IF ('æ' COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS =
* 'ae' COLLATE SQL\_Latin1\_General\_CP1\_CI\_AS)
* BEGIN
* PRINT 'SQL\_Latin1\_General\_CP1\_CI\_AS';
* END;
* IF ('æ' COLLATE Latin1\_General\_100\_CI\_AS =
* 'ae' COLLATE Latin1\_General\_100\_CI\_AS)
* BEGIN
* PRINT 'Latin1\_General\_100\_CI\_AS';

END;

Returns:

Latin1\_General\_100\_CI\_AS

The only thing you are "losing" here is *not* being able to do these expansions. Generally speaking, this is another benefit of moving to a Windows Collation. However, just like with the "String Sort" to "Word Sort" move, the same caution applies: it is a definite difference of behavior between the two collations (again, just for VARCHAR data), and you might have code and/or customer expectations based upon *not* having these mappings. **This requires testing your code and possibly researching to see if this change in behavior might have any negative impact on users.**

(first noted in @Zarepheth's[answer](https://stackoverflow.com/a/22055580/577765)and expanded on here)

* Another difference (that is also a benefit of moving to a Windows Collation) is that filtering a VARCHAR column that is indexed on NVARCHAR literal / variable / column you will no longer invalidate the index on the VARCHAR column. This is due to the Windows Collations using the same Unicode sorting and comparison rules for both VARCHAR and NVARCHAR data. Because the sort order is the same between the two types, when the VARCHAR data gets converted into NVARCHAR (explicitly or implicitly due to datatype precedence), the order of items in the index is still valid. For more details on this behavior, please see my post: [Impact on Indexes When Mixing VARCHAR and NVARCHAR Types](https://sqlquantumleap.com/2017/07/10/impact-on-indexes-when-mixing-varchar-and-nvarchar-types/).
* The server-level Collation is used to set the Collation of the system databases, which includes [model]. The [model] database is used as a template to create new databases, which includes [tempdb] upon each server startup. So, if the Database's default collation does not match the instance's default Collation *and* you join local tables to temporary tables on string fields, then you will get the Collation-mismatch error. Fortunately there is a somewhat easy way to correct for collation differences between the database that is "current" when CREATE #TempTable is executed and [tempdb]. When creating temporary tables, declare a collation (on string columns) using the COLLATE clause and use either a specific collation (if you know that the DB will always be using that collation), or DATABASE\_DEFAULT (if you don't always know the collation of the DB where this code will execute):

CREATE TABLE #Temp (Col1 NVARCHAR(40) COLLATE DATABASE\_DEFAULT);

This is not necessary for table variables since they get their default Collation from the "current" database. However, if you have both table variables and temporary tables and join them on string fields, then you will need to use COLLATE {specific\_collation} or COLLATE DATABASE\_DEFAULT as shown directly above.

* The server-level collation also controls local variable names, CURSOR variable names, and GOTO labels. While none of these would be impacted by the specific change being dealt with in this Question, it is at least something to be aware of.
* It is best to use the most recent version of the desired collation, if multiple versions are available. Starting in SQL Server 2005, a "90" series of collations was introduced, and SQL Server 2008 introduced a "100" series of collations. You can find these collations by using the following queries:
* SELECT \* FROM sys.fn\_helpcollations() WHERE [name] LIKE N'%[\_]90[\_]%'; -- 476

SELECT \* FROM sys.fn\_helpcollations() WHERE [name] LIKE N'%[\_]100[\_]%'; -- 2686

* ALSO, while the question asks about case-insensitive Collations, it should be noted that if someone else is looking to make a similar change but is using case-sensitive Collations, then another difference between SQL Server Collations and Windows Collations, *for VARCHAR data only*, is which case sorts first. Meaning, if you have both A and a, the SQL\_ Collations will sort A before a, while the non-SQL\_ Collations (and the SQL\_ Collations when dealing with NVARCHAR data) will sort a before A.

For a lot more info and details on changing the Collation of a Database or of the entire Instance, please see my post:  
[Changing the Collation of the Instance, the Databases, and All Columns in All User Databases: What Could Possibly Go Wrong?](https://sqlquantumleap.com/2018/06/11/changing-the-collation-of-the-instance-and-all-columns-across-all-user-databases-what-could-possibly-go-wrong/)

For more info on working with strings and collations, please visit: [Collations Info](https://collations.info/)

SELECT top 1000 DB\_ID(tab.TABLE\_CATALOG) DatabaseId

, tab.TABLE\_CATALOG

, tab.TABLE\_SCHEMA

, tab.TABLE\_NAME

, tab.TABLE\_TYPE

, systab.type AS Table\_Type

, systab.type\_desc AS Table\_TypeDescription

, syscols.name AS Column\_Name

, tp.name AS SysTypeName

, CASE

WHEN tp.[name] IN ('varchar', 'char') THEN tp.[name] + '(' + CAST(syscols.max\_length AS NVARCHAR) + ')'

WHEN tp.[name] IN ('nvarchar', 'nchar') THEN tp.[name] + '(' + CAST(syscols.max\_length AS NVARCHAR) + ')'

WHEN tp.[name] IN ('decimal', 'numeric') THEN tp.[name] + '(' + CAST(syscols.[precision] AS VARCHAR(25)) + ', ' + CAST(syscols.[scale] AS VARCHAR(25)) + ')'

WHEN tp.[name] IN ('datetime2') THEN tp.[name] + '(' + CAST(syscols.[scale] AS VARCHAR(25)) + ')'

ELSE tp.[name]

END

--, CASE

-- WHEN tp.[name] IN ('varchar', 'char') THEN tp.[name] + '(' + IIF(syscols.max\_length = -1, 'max', CAST(syscols.max\_length AS VARCHAR(25))) + ')'

-- WHEN tp.[name] IN ('nvarchar','nchar') THEN tp.[name] + '(' + IIF(syscols.max\_length = -1, 'max', CAST(syscols.max\_length / 2 AS VARCHAR(25)))+ ')'

-- WHEN tp.[name] IN ('decimal', 'numeric') THEN tp.[name] + '(' + CAST(syscols.[precision] AS VARCHAR(25)) + ', ' + CAST(syscols.[scale] AS VARCHAR(25)) + ')'

-- WHEN tp.[name] IN ('datetime2') THEN tp.[name] + '(' + CAST(syscols.[scale] AS VARCHAR(25)) + ')'

-- ELSE tp.[name]

--END

, syscols.collation\_name AS Column\_Collation

, DATABASEPROPERTYEX(tab.TABLE\_CATALOG, 'Collation') AS Database\_Collation

, SERVERPROPERTY('collation') AS SQLServer\_Collation -- Collation used by the current SQL Server instance

FROM INFORMATION\_SCHEMA.TABLES tab

inner join sys.tables systab

on tab.TABLE\_NAME = systab.[name]

inner join sys.columns syscols

on syscols.object\_id = systab.object\_id

left outer join sys.types tp on tp.system\_type\_id = syscols.system\_type\_id;

--WHERE syscols.collation\_name = 'SQL\_Latin1\_General\_CP1\_CI\_AS'

--AND tab.TABLE\_CATALOG = 'NavPlus';

SELECT SERVERPROPERTY('collation') AS SQLServer\_Collation;

Select DATABASEPROPERTYEX(tab.TABLE\_CATALOG, 'Collation') as Database\_Collation -- collation database

From INFORMATION\_SCHEMA.TABLES tab

Group BY DATABASEPROPERTYEX(tab.TABLE\_CATALOG, 'Collation');

Select syscols.collation\_name

, COUNT(1) AS Amount-- collation columns

From sys.columns syscols

inner join sys.tables systabs

on systabs.object\_id = syscols.object\_id

WHERE syscols.collation\_name<>''

Group BY syscols.collation\_name

return

Select systabs.name ,syscols.collation\_name

, COUNT(1) AS Amount-- collation columns

From sys.columns syscols

inner join sys.tables systabs

on systabs.object\_id = syscols.object\_id

WHERE syscols.collation\_name<>''

Group BY systabs.name, syscols.collation\_name

/\*

You should see little difference if the collation is SQL\_Latin1\_General\_CP1\_CI\_AS or Latin1\_General\_CI\_AS, but both have instances where they are faster or slower than the other.

Latin1\_General\_CI\_AS :- Latin1-General, case-insensitive, accent- sensitive, kanatype-insensitive, width-insensitive

SQL\_Latin1\_General\_CP1\_CI\_AS :- Latin1-General, case-insensitive, accent-sensitive, kanatype-insensitive, width-insensitive for Unicode Data, SQL Server Sort Order 52 on Code Page 1252 for non-Unicode Data

\*/