



**Using Collations with GUPTA[®] SQLBase[®]
International**

By Michael Vandine - Senior Technical Support Consultant

January 2006

GUPTA[™]

Abstract.....	3
Introduction	3
What are Collations?	4
Binary Collation	4
Additional Collations with Sensitivities.....	4
How Does the Locale Affect Collations?	5
How Do You Choose a Collation to Use?.....	6
How are Collations Used with SQLBase?.....	6
Defined Collation Levels and Precedence.....	7
Changing Collations.....	7
Referential Integrity Considerations	8
Performance Considerations.....	8
View and Stored Command Considerations	8
Collations Versus NLS Sorting	9
An Example of CS/CI and AS/AI Collation Results.....	9
Collation Statement Examples	10



SQLBase International supports a multitude of collations rather than just a binary collation.

Abstract

This paper explains the concepts of collations and how they are best used in SQLBase International.

Introduction

In earlier versions of SQLBase it was impossible to specify a collation, a comparison or ordering of data honoring a specific locale. The collation used was automatically BINARY. Beginning with SQLBase International, there are a multitude of additional collations supported, with binary still being one of the choices available. To use these additional collations, it is important to understand what a collation is, how to select the correct collation and the ramifications of using collations.

Collation is the comparison or ordering of data honoring a specific locale

What are Collations?

Collation is defined as the rules that are applied when characters are compared to each other, honoring a specific locale. Collation is used in any SQL statement that involves the comparison of strings.

A locale identifies a specific user community – a group of users who have similar culture and language expectations for human-computer interaction and the kinds of data they process. Collation is dependant upon the context of data use, so the explicit declaration of a collation and/or locale is required to provide the context that a user requires.

Binary Collation

The most basic type of collation is the binary collation. This is where you take the numeric value of the character and you compare it with the numeric value of another character. In a binary collation, a=0x61, b=0x62, A=0x41 and B=0x42, so if we order them, they will be ordered A, B, a, b.

But if you look in a dictionary, you will notice that the words are not in that order. The words are usually in a, A, b, B order or ordered without regard to case.

SQLBase International supports a binary collation and if you want SQLBase to behave as it did in previous versions you should not change collations.

Additional Collations with Sensitivities

SQLBase International supports many more collations. These additional collations are dictionary sorts with options regarding Case and Accent sensitivity.

Let's extend the first example by adding a bit more data. Consider the following data: a, aa, aA, A, AA, Aa, b, bA, ba, B, BA, Ba. With a binary collation the database would be ordered as A, AA, Aa, B, BA, Ba, a, aA, aa, b, bA, ba.

But suppose you weren't concerned about upper-case versus lower-case. With a case insensitive Dictionary sort, the data **could** be in the following order: A, a, AA, aA, aa, Aa, b, B, Ba, bA, ba, BA. They **could** also be in the order: a, A, aa, aA, AA, Aa, B, b, ba, BA, Ba, bA. Note the word 'could'. There are additional permutations, but think of it this way: a case insensitive ordering means that the characters a and A are not only equal, but the same. With that being true, a=A, aa=aA=AA=Aa and b=B, ba=BA=bA=Ba. Since these equalities hold true, there is no guarantee to the order of equalities.

If this is confusing, consider that if you have a, a, a, a and order them, which one is first? You really don't know because they are dependent upon the 'stability' of the sort.

Now consider the addition of the following characters: á, â, a, ä. Where do these characters fit in the picture? Well this depends upon the addition of Accent Sensitivity or Insensitivity. Accent Sensitivity indicates there is a difference between those characters and 'a' but Accent Insensitivity indicates that they are all equal to 'a'.

How Does the Locale Affect Collations?

Different languages and even more subtly different locales of the same language order/compare characters differently:

- The letters A-Z can be sorted in a different order than in English. For example, in Lithuanian, "y" is sorted between "i" and "k".
- Combinations of letters can be treated as if they were one letter. For example, in traditional Spanish "ch" is treated as a single letter, and sorted between "c" and "d".
- Accented letters can be treated as minor variants of the unaccented letter. For example, "é" can be treated equivalent to "e".
- Accented letters can be treated as distinct letters. For example, "Å" in Danish is treated as a separate letter that sorts just after "Z".
- Unaccented letters that are considered distinct in one language can be indistinct in another. For example, the letters "v" and "w" are two different letters according to English. However, "v" and "w" are considered variant forms of the same letter in Swedish.
- A letter can be treated as if it were two letters. For example, in traditional German "ä" is compared as if it were "ae".
- Thai requires that the order of certain letters be reversed.
- French requires that letters sorted with accents at the end of the string be sorted ahead of accents in the beginning of the string. For example, the word "côte" sorts before "coté" because the acute accent on the final "e" is more significant than the circumflex on the "o".
- Sometimes lowercase letters sort before uppercase letters. The reverse is required in other situations. In case sensitive collations, uppercase is sorted before lowercase and SQLBase does not have an option to change this behavior.
- Even in the same language, different applications might require different sorting orders. For example, in German dictionaries, "öf" would come before "of". In phone books the situation is the exact opposite.
- Sorting orders can change over time due to government regulations or new characters/scripts in Unicode.



How Do You Choose a Collation to Use?

SQLBase International uses a product from IBM called ICU. Initially the collations to select will be limited to those that ICU provides.

The available collations to choose from can be found in the system table syscollations

To see the available collations that SQLBase supports, in SQLTalk you can do a `SELECT * FROM SYSADM.SYSCOLLATIONS;` to see them all. In most cases they are broken down by language or script with some options. For example, South America speaks Spanish but they order differently than Spain. Thus South America would use a `Spanish_Traditional` collation instead of a Spanish collation.

The structure of the Collation Name is:

Language_Regional Option_Case Sensitivity_Accent Sensitivity

If you do a `"select * from syscollations where name like 'Spanish%';"` you will see:

- 9 Spanish_tradtional_CS_AS
- 10 Spanish_tradtional_CS_AI
- 11 Spanish_tradtional_CI_AS
- 12 Spanish_tradtional_CI_AI
- 74 Spanish_CS_AS
- 75 Spanish_CS_AI
- 76 Spanish_CI_AS
- 77 Spanish_CI_AI

You can then choose the appropriate collation based on if you want case or accent sensitivity or not. For example, if you want case sensitivity on but not accent sensitivity, you would choose the collation `Spanish_tradtional_CS_AI`.

How are Collations Used with SQLBase?

Collations are supported at various levels. It is very important that these levels and the precedence that SQLBase uses when determining the collation for a comparison are understood. These levels are defined as either explicit (specifically defined) or implicit (inherited from the value of a lower level). To put it simply, the collation used will always be the highest **explicitly** defined collation.

Defined Collation Levels and Precedence

The collation that is used is always the highest level explicitly defined collation

Level 1 - Defined at the server level - All SQLBase servers have a collation. This value is **always** considered explicit and can be specified in the server's sql.ini file with a 'collation' statement. If unspecified the value is binary.

Level 2 - Defined at the database level - A database may have a collation defined at creation time. If it does, then the value is considered explicit. If it doesn't then the value is considered implicit. The implied value is the value defined at the server.

Level 3 - Defined at the session level - Each session may have a collation defined set with a 'set collation' statement or by an API call. If it does, then the value is considered explicit. If it doesn't then the value is implicit and the implied value is the value of the database.

Level 4 - Defined at the table level - A collation can be defined for a table at creation time. If it does, then the value is considered explicit. If it doesn't then the value is implicit and the implied value is that of the session.

Level 5 - Defined at the column level - A collation can be defined for a column within a table at creation time. If it does, then the value is considered explicit. If it doesn't then the value is implicit and the implied value is that of the table.

Level 6 - Defined at the statement level - For each select list column (in a select) and every predicate comparison a collation can be specified. If it does, then the value is considered explicit. If it doesn't then the value is implicit and the implied value is that of the column.

For indexes, the collation value cannot be implied. Character columns in an index are always explicit. If you don't specify them in the creation of the index then their explicit value is taken in the precedence order from column up to server levels. Please note that if you want to change the collation of an existing column in an index, the index should probably be dropped. The reason for this is the need to construct appropriate indexes that have the keys ordered correctly. Consider that you have a class of users that use English_CS_AS collation and a class of users that use German_Phonebook collation. There needs to be access paths (effective indexes) for both collations.

Changing Collations

The collations already defined for a database can be changed. This is an extremely powerful function, since it allows the internationalization of a database from one language or locale simply by changing the collation value for the whole server or the individual database. These changes can be made at each of the levels defined above to make the changes permanent, (ie. with an 'alter database' command or 'alter table' command) or by specifying the new collation for a session or query.



**Collations
already defined
can be changed
to easily allow for
changes to
language or
locale**

However, care must be taken when changing the collations, since indexes defined will retain the original collation (remember that they are always defined as 'explicit') and they may cause a performance problem since they may not be used with the new collation definition. Indexes will most likely need to be dropped and rebuilt.

There could also be a problem with different results being returned based on the Case and Accent Sensitivity in the newly defined collation.

Referential Integrity Considerations

Primary key indexes are system generated with the collation of the index depending on the collation precedence hierarchy. Just as an index always has an explicit collation defined, a primary key also has an explicit collation. This means that the collation on the primary key cannot be changed. The implications of this is that any foreign key columns that are defined must also use the same collation as the matching primary key and, to ensure RI enforcement, the collation of these foreign key columns also cannot change.

You must be very careful when selecting what Case and Accent Sensitivity is defined for RI. For example with a CI_AI (Case and Accent Insensitivity) defined, $a=A=Å$ which could lead to some unexpected results for the enforcement of RI.

Performance Considerations


Consider the differences between a binary collation and a dictionary ordering collation. A binary compare is a simple memory comparison and requires a relatively small amount of instructions to complete. However, with different collations, much more logic must be performed to compare the two characters.

The end result is that a collation comparison requires many more internal instructions than a binary comparison and thus a binary collation will **always** outperform a non-binary collation.

Multiple indexes can be created on the same column/columns with different collations. This will improve the performance of the query when a query is executed to fetch results with different collations (explicit or overriding with session collations).

View and Stored Command Considerations

Views are dynamic statements that honor the collation nature of the underlying columns, tables, session, database or server. Therefore, if the collations are implicitly set, changing collations at any of the lower precedence levels may affect the query results of the view, based upon comparisons or grouping.



Stored commands, however, are pre-built commands, with the access path already decided by the optimizer based on the collation settings when the stored command was built. If a collation is changed globally, the execution plan would not change and an index with an explicit collation that might not match the new collation could be used. Therefore, extra caution must be used when changing collations so that performance problems are avoided. It is probably wise to recreate the indexes used by the stored command to explicitly set the collation, then drop and re-create the stored command.

Collations Versus NLS Sorting

In previous versions of SQLBase, if you wanted to change the sorting order you had to set up a specific country.sql file that contained a #translate section where you defined what order you wanted to use for your sorting. The major disadvantage with this was the requirement that all databases and sorting was then locked into that one order. With collations, you are no longer locked into one sort order for all databases on the server, nor are you limited to the sorting even within a single database. This provides complete flexibility in ordering.

A CS/CI and AS/AI example

An Example of CS/CI and AS/AI Collation Results

The selection of the Case and Accent Sensitivity not only affects the ordering of the data, it also impacts on the results returned from a SQL statement.

Comparisons are directly affected by the collation selected. For example, if you have a column that is defined as CI_AI and there are col1 column values of 'a', 'A', and 'Ä', a statement like:

```
select * from table1 where col1 = 'A'
```

will return all three rows. If a collation is defined with CI_AS, the same select would return the values 'a' and 'A'. If a collation is defined with CS_AI, the same select would return values 'A' and 'Ä'.

This will have the same affect when doing a select with a 'group by' column. The grouping will be done based on the Case and Accent Sensitivity.



Collation Statement Examples

Collation statements can be used just about anywhere to define or override previously defined collations.

Explicit collation happens when the COLLATE keyword is part of a specific query.

```
SELECT CHINESE_COLUMN, STANDARD_COLUMN FROM MY_TABLE  
WHERE STANDARD_COLUMN COLLATE LATIN_GENERAL_CS_AS > 'G';
```

Collation statement examples

They can also be used in a 'group by' and 'order by' clause to override the currently defined collation just by specifying 'COLLATE collation' after the normal grouping or sorting syntax.

Implicit collation is assigned at the time an object is created. Three examples:

```
CREATE DATABASE MY_DB COLLATE English_CI_AI;  
CREATE TABLE MY_TABLE ( column declarations ) COLLATE  
English_CI_AI;  
CREATE TABLE MY_TABLE ( CHINESE_COLUMN NCHAR(1000) COLLATE  
GB2312, STANDARD_COLUMN CHAR(20) ) COLLATE English_CI_AI;
```

Server collation can be assigned using the COLLATE keyword in the server portion of the configuration file (sql.ini). Client collation can be assigned using the COLLATE keyword in the client portion of the configuration file (sql.ini).

Collations can ALSO be changed with the 'Alter Table' command:

```
ALTER TABLE MY_TABLE COLLATE English_CI_AI;  
ALTER TABLE MY_TABLE MODIFY STANDARD_COLUMN COLLATE  
English_CI_AI;
```