Sorting Your Linguistic Data inside an Oracle Database

An Oracle Technical White Paper
May 2005

ORACLE

Sorting Your Linguistic Data inside an Oracle Database

Sorting Your Linguistic Data inside an Oracle Database

INTRODUCTION

Sorting character strings can be an extremely intricate operation, the complexity of which may not be apparent to most users. Different languages have their own sorting rules, some languages are collated according to the letter sequence in the alphabet, some according to the number of stroke counts in the letter, and there are even languages which are ordered by the pronunciation of the words. Treatment of letter accents differs among languages as well.

Sorting is not straightforward for day-to-day English usage either. If you look up words in an English dictionary, you will probably find that upper and lower case characters are mixed together in terms of ordering. Looking for names in the telephone directory, you may find that certain words may be treated the same, for example the prefix Mac and Mc are grouped together.

Sorting can be further complicated when you need to sort data from more than one language - how do you handle the case when one language demands that a given letter is sorted after the letter *z*, while the same letter in another language requires it to be collated before the letter *a*?

In the rest of this paper we will use the term 'linguistic sort' or collation to describe the culturally expected ordering of elements of a text list if this list is considered sorted in a given language. A native speaker of the language expects to locate an element of the list relative to other elements based on this ordering. For example, English speaking users would expect to find a word beginning with B to come after all words starting with A and before all words beginning with C in an ordered list of English words.

Presenting data not sorted in the linguistic sequence that your users are accustomed to can make searching for information difficult and time consuming.

In Oracle Database 10*g*, there is comprehensive coverage of binary sorts, linguistic sorts and multilingual linguistic sorts to meet the demands of customers who need to search and sort data in multiple languages. Case insensitive and accent insensitive modes are new features that offer even finer control. This paper outlines the basic concept of linguistic sort, provides examples of how sort properties influence the

collation order of different languages, and explains how you can customize the way your data is sorted inside an Oracle database.

THE WORLD OF SORTING RULES

The section above mentioned some of the intricacies involved when sorting data in a culturally expected order. This section will go over them in more detail showing that every writing script (e.g. Latin, Greek, Cyrillic etc.) has an ordering of its own and some writing scripts have many conflicting orderings for the different languages that they support.

Western European Languages

In German, *ß* is one character, but it is treated as double s '*ss*' for ordering purposes. German speakers are also accustomed to treating *ü*, *ä* and *ö* as equivalent to *ue*, *ae* and *oe* respectively. These three characters are sorted in the same order as the vowel pairs.

Spain traditionally treats *ch*, *ll* as well as *ñ* as letters of their own, ordered after *c*, *l* and *n* respectively. For example, the following Spanish words would be sorted as listed:

```
cabalmente, caballa, cantina, caña, clamar, curador, chácara
```

Recently, this traditional Spanish sorting practice has been replaced with the modern Spanish sort, which removes the special status of *ch* and *ll*.

In the Danish alphabet, three additional letters namely *æ*, *ø* and *å* sort after *z*. It also treats the letter combination *aa* as a separate letter ordered last in the alphabet after *å* and also considers letters *ü* and *y* as variants of the same letter. Due to these rules, the following cities Zürich, Aarfit and Århus will all appear after Zyrardow in a list of cities.

In French, accented vowels are first treated like unaccented vowels from the point of view of their base sorting order. After the text is sorted without considering accents, it is further sorted so that within each vowel set, the order is no accent, acute accent, grave accent, circumflex accent followed by umlaut. To make this even more complex, accented vowels are evaluated from right to left, while non accented base characters are weighted from left to right. Thus, *Èdít* comes before *Edít* when collating the two strings using a French linguistic sort.

Accented letters introduce other complexity as they can have different meanings across languages and sometimes even within the same language. Complication arises when dealing with multi-national coding schemes such as Unicode that

support several languages, since these languages may contain conflicting alphabetic rules for the same letter. For example, the letter *ä* (a with an umlaut) is sorted before *b* in German but it is sorted after *z* in Swedish. Thus, if the sorted text belongs to multiple languages it usually cannot be sorted correctly from the point of view of all of them.

Asian Ideographs

As you can see, cultural expectations vary a great deal between languages and the previous examples are just for Western European languages using the same Latin writing script. If dealing with 26 to 40 letters in an alphabet is quite a challenge, then working with Asian ideographs can be an extremely daunting task. For the Chinese language alone, there are over 40,000 characters. The sorting rules also vary between Simplified (used in China and Singapore) and Traditional (used in Taiwan, Hong Kong & Macau) Chinese writing scripts.

Here are some examples of the most common collation methods used for sorting Chinese characters.

Stroke count - This is one of the most common ways to sort Chinese characters. Chinese characters are composed of a radical (base element) and zero or more components. Both the radical and the components are composed of strokes. The number of strokes varies between characters. Characters with identical stroke counts will be further sorted by the radical.

This example illustrates how three Chinese characters are ordered based on their stroke counts.

串(stroke: 7, radical: |) < 吟(stroke:7, radical:口) <

鼗(stroke: 17, radical:土)

The number of stroke counts for a given Chinese character may differ between its simplified and traditional form. Hence, it is common to have two distinct sorting algorithms to handle both writing styles.

Radical - This ordering is found in most Chinese dictionaries. It is similar to the stroke count ordering but here the radical ordering takes priority over the stroke count. Although it is a more traditional method of sorting Chinese characters, sometimes it can be quite cumbersome even for native Chinese speakers since it may not be intuitive to locate the correct radical for a given Chinese character. The order of the radical is by the radical stroke count. For characters with identical radical order, the number of stroke counts of the character will be used.

This example illustrates how three Chinese characters are ordered based on their radical stroke counts.

土(radical:土,stroke:3) < 地(radical:土,stroke:6) <

石(radical:石,stroke:5)

Just like in the previous stroke count sort, some Chinese characters may have a different variation of the radical between the simplified and traditional form of Chinese. Hence, it is common to have two distinct sorting algorithms to handle the differences between the two writing styles.
Pronunciation (Pinyin) – Pinyin is the pronunciation based sort for Simplified Chinese, Bopomofo is its counterpart for Traditional Chinese. Pronunciation sort sounds straightforward until you encounter words that can pronounce differently depending on the context. There are also cases where characters with the same Pinyin can have five different tones; there are four tones in spoken Chinese plus a special light tone. Tones are marked from 1 to 5, with the light tone being the fifth tone and the last in the order of tones. Characters with the same Pinyin and tone are further sorted by the stroke count.

The example below illustrates how four Chinese characters are ordered based on their Pinyin sequence. Names in parentheses show the Pinyin and the tone values of these characters.

阿(A1) < 伯(Bo2) < 蔔(Bo5) < 出(Chu1)

ISO/IEC 14651 – INTERNATIONAL STRING ORDERING

In the mid-1980s a Québeccer , Alain LaBonté was surprised and puzzled with the inconsistent behavior with which the different commercial sort programs dealt with French homograph words, even though these were intuitive to French native speakers. He realized the need for an international standard that would allow definition of a universal methodology for multi-script ordering.

In 1992, the project for an international string ordering standard - ISO 14651 was created. This international standard provides a method for ordering text data worldwide, and provides a Common Template Table whose tailoring meets the requirements of a given language and culture while retaining universal properties for other scripts.

The Common Template Table requires some tailoring in different local environments. However, conformance to this International Standard requires that all deviations from the Template, called "deltas", be declared to document result discrepancies. This Standard describes a method to order text data independently of context.

SORTING INSIDE THE ORACLE DATABASE

Text is conventionally sorted inside a database according to the binary codes used to encode the characters. Typically, this does not produce a sort order that is linguistically correct. In some cases, it can be correct if the given encoding scheme specifies all the characters in ascending binary value according to the appropriate

alphabetic convention. Unfortunately, most encoding schemes do not follow any such convention, and even if they do, it is not possible for them to cover the more complex sorting scenarios discussed previously.

To overcome this limitation, Oracle provides linguistic sorting. A Linguistic sort handles the complex sorting requirements of the different languages and cultures. It enables text in any character encoding schemes to be sorted according to specific linguistic conventions, independent of the binary values of the characters.

Three types of database sorts are supported in Oracle database 10*g*:

1. Binary sort
2. Monolingual linguistic sort
3. Multilingual linguistic sort

Binary Sort

The most common way to sort character data is to order them by their numeric binary codes as defined by the character encoding scheme. This is achieved by using Binary sort. It is the fastest form of sorting in the database because no special processing has to be done on sorted values. Standard database indexes are always based on the binary codes, they are usually smaller, requiring less disk reads to search than linguistic indexes, described later in this paper.

Binary sort offers reasonable results for the English alphabet since both ASCII and EBCDIC standards define the letters from A to Z in ascending numeric value order. However, this is not perfect since uppercase letters and lowercase letters are grouped separately. For ASCII, uppercase letters appear before any lowercase letters, whereas in EBCDIC it is the reverse.

Results generated from binary sorts can vary depending on the ordering of the characters within the given character sets. When characters used in languages other than English are present, binary sorts usually do not produce reasonable results. For example, an ascending ORDER BY query returns the characters C, E, b, Ë, â, because each preceding character has a lower numeric code in the character encoding scheme than the one after it. A Binary sort cannot present linguistically meaningful data when the language sorting rules are complex.

The Oracle binary sort is named BINARY. Oracle also supports the five quasi-binary sorts: UNICODE_BINARY, ASCII7, EBCDIC, BIG5, GBK, and HKSCS. These sorts are binary in that they sort data according to binary codes of characters in AL16UTF16, US7ASCII, WE8EBCDIC1047, ZHT16BIG5, ZHS16GBK, and ZHT16HKSCS character sets respectively. However, they are not equivalent to the

BINARY sort because the sorted strings have to be converted to the correct character set before comparison. This means that standard BINARY indexes cannot be used to satisfy queries for data ordered by these sorts and linguistic indexes have to be defined.

Monolingual Linguistic Sort

To produce a 'localized' sort sequence that adheres to a specific linguistic convention, another sort technique must be used that can sort characters independently of their binary codes inside the character-encoding scheme. This technique is called a linguistic sort. A linguistic sort operates by replacing characters with numeric values also known as 'sort keys' that reflect each character's proper linguistic order for a given language.

The construction of the sort key is achieved by breaking down each letter of the alphabet into two components: major value and minor value. Usually, letters with the same appearance (or base letter) have the same major value; minor value is used to differentiate diacritic and case variants of the same base letter.

The following table shows sample letters and their major and minor values:

Letter	Major value	Minor value
a	10	5
A	10	10
ä	10	15
Ä	10	20
B	20	5

Character strings are compared in two steps for monolingual linguistic sorts. First, the algorithm generates sort keys for the compared values and than it compares the sort keys byte by byte until they differ or one is shorter. The shorter key or the one with the lower differing byte is considered smaller.

The sort keys are generated by concatenating all major sort values of all characters of the strings followed by a zero value and then by all minor values. This way the major values determine the base sort order and only for strings with the exactly same major values, the minor values are compared to further order the strings.

If a character, like space or hyphen, should not be considered when comparing strings, then only the minor value is used for the sort key. The minor value is not

skipped so that strings differing by ignorable characters only are not considered equal.

The result of the SQL NLSSORT function is the sort key in the RAW datatype.

Oracle uses named linguistic sorts to specify how character data should be sorted. The names of the linguistic sorts in most cases are the same as the language names. For some languages, additional 'extended' linguistic sorts are defined; these linguistic sorts are named with a preceding 'X'.

For example, Oracle supports two Spanish monolingual linguistic sorts, `SPANISH` for Modern Spanish and `XSPANISH` for Traditional Spanish.

Extended linguistic sorts are designed to accommodate language-specific special cases:

1. Sorting of digraphs as a single character, e.g. the Spanish *ll* and *ch*.
2. Converting single characters into digraphs for sorting purposes, e.g. the German sharp s '*ß*' is treated as *ss*.

The following is a complete list of monolingual sorts supported in Oracle Database 10*g*. Bolded entries indicate new sorts introduced in Oracle Database 10*g*.

ARABIC
ARABIC_MATCH
ARABIC_ABJ_SORT
ARABIC_ABJ_MATCH
AZERBAIJANI
XAZERBAIJANI
BENGALI
BULGARIAN
CATALAN
XCATALAN
CROATIAN
XCROATIAN
CZECH
XCZECH
CZECH_PUNCTUATION
XCZECH_PUNCTUATION
DANISH
XDANISH
DUTCH
XDUTCH
EEC_EURO
EEC_EUROPA3
ESTONIAN
FINNISH
FRENCH
XFRENCH
GERMAN
XGERMAN
GERMAN_DIN
XGERMAN_DIN
GREEK
HEBREW
HUNGARIAN
XHUNGARIAN
ICELANDIC
INDONESIAN
ITALIAN
LATIN
LATVIAN
LITHUANIAN
MALAY
NORWEGIAN
POLISH
PUNCTUATION
XPUNCTUATION
ROMANIAN
RUSSIAN
SLOVAK
XSLOVAK
SLOVENIAN
XSLOVENIAN
SPANISH
XSPANISH
SWEDISH
SWISS
XSWISS
TURKISH
XTURKISH
UKRAINIAN
VIETNAMESE
WEST_EUROPEAN
XWEST_EUROPEAN

You can specify any of the above linguistic sorts irrespective of whether the data is stored in ASCII or EBCDIC encoding form, as linguistic sorts are independent of the actual character-encoding scheme.

Multilingual Linguistic Sort

Monolingual linguistic sort is useful when you are comparing and sorting data in one language, however it cannot process data across multiple languages or writing systems. Multilingual linguistic sorts allow you sort data in more than one language in a single sort. With the focus on data consolidation, it is becoming common for customers to have different language data in a single global database. Multilingual linguistic sort helps customers with multilingual data to accurately search and organize information in any language.

Oracle's multilingual linguistic sort is based on the ISO/IEC 14651 and the Unicode Collation Algorithm standards. For it to handle the complex multilingual sorting requirements and at the same time to provide a greater flexibility for modification, multilingual linguistic sorts are evaluated in three levels of precisions, as defined by the above two standards.

1. Primary level – A primary level sort distinguishes between base characters, such as the difference between characters *a* and *b*. If a character is an ignorable character, then it is assigned a primary level order (or weight) of zero, which means it can be ignored during sorting comparison at this particular level. An example of an ignorable character is the dash character '-'. If it is ignored, then the word '*multi-lingual*" can be treated the same as *'multilingual*'.
2. Secondary level – this is used to distinguish between the different diacritics for a given base character. For example, the character *Ö* differs from the character *O* only because it has a diacritic. Thus, *Ö* and *O* are the same on the primary level because they have the same base character *O* but differ on the secondary level. The secondary level may be specified as considering characters starting from the end of the string toward its beginning. This is required for the French sort order.
3. Tertiary level – A tertiary level sort distinguishes between casing (upper and lower case) of characters that do not differ on the primary and secondary levels.

The following example illustrates how a list is sorted based on all three levels. At the primary level 'resume' is ordered before 'resumes'; at the secondary level, strings without diacritics come before strings with diacritics and at the tertiary level, lower case characters are sorted before upper case ones.

```
resume
Resume
résumé
Résumé
resumes
```

This three level architecture enables the Oracle database to handle languages that have complex sorting rules, as well as providing linguistic support for databases with multilingual data. Linguistic sorts for Asian languages such as Chinese, Japanese and Korean are now available for the first time. In fact, for Chinese sorts, we now offer three varieties of sorts based on the number of strokes, Pin Yin and radicals.

In addition, multilingual linguistic sorts also provide 'canonical equivalence' and 'supplementary characters' support.

One Unicode character may be equivalent to a sequence of base character plus combining characters. This is called 'canonical equivalence'. For example, the character *ä* is equivalent to the combination of the base letter *a* follow by a combining character diaeresis ¨. According to the Unicode standard, canonical equivalent strings should be sorted as being equal; for this reason, strings are put into their normalized form before any linguistic sensitive comparison takes place. Supplementary characters are the newly defined characters introduced in Unicode 3.1 standard that require four bytes for storage; potentially using supplementary characters an additional one million characters can be encoded into the Unicode standard. Currently there are less than 100,000 characters defined in the latest Unicode standard 3.2. With the new Multilingual linguistic sort architecture in Oracle, up to 1.1 million characters can be defined in one sort, covering all the characters that can be added to the Unicode standard.

Based on ISO/IEC 14651 – International String Ordering, Oracle provides a common multilingual collation (sorting) template called `GENERIC_M`. `GENERIC_M` provides coverage to Latin, Cyrillic and Greek based languages; it defines accent and punctuation characters at the beginning of the sort definition. Uppercase and lowercase letters are grouped together with the lowercase version of each base character appearing before uppercase one. It also covers the grouping of numbers including full-width numbers and roman numeric letters. There are around 1,000 characters defined in the `GENERIC_M` template; characters that are not defined are sorted according to their Unicode binary order.

Oracle's multilingual linguistic sorts are all created based on the `GENERIC_M` template; their names are post fixed with '`_M`' to indicate that they are multilingual sorts based on the ISO 14651 standard.

The following is a complete list of multilingual sorts supported in Oracle.

GENERIC_M	JAPANESE_M	SPANISH_M
	KOREAN_M	TCHINESE_RADICAL_M
CANADIAN_M	SCHINESE_STROKE_M	TCHINESE_STROKE_M
DANISH_M	SCHINESE_PINYIN_M	THAI_M
FRENCH_M	SCHINESE_RADICAL_M	

For example, Oracle supports a monolingual French sort called 'XFRENCH'. You can also achieve the same behavior by specifying the multilingual French sort called 'FRENCH_M' but this sort is based on the GENERIC_M sorting order and can sort diacritical marks from right to left as required in the French language. The benefit of the multilingual sort is that it can order non-French specific characters in an internationally recognized sequence as defined by the ISO 14651 standard.

In general, using Multilingual linguistic sort is recommended; regardless of the number of languages that are stored in the database. This is a standard based implementation, going forward it will always be enhanced to support latest changes in the standard.

Case and Accent Insensitive Searching in Oracle Database 10g

Operations inside an Oracle database are sensitive to the case and the accents (diacritics) of the characters. Sometimes you may need to perform case-insensitive or accent-insensitive comparisons and sorts. In previous releases, you could call SQL LOWER and UPPER functions to make SQL statements case-insensitive or use the GENERIC_BASELETTER sort. Using the SQL LOWER and UPPER functions can be expensive, because they convert every character in the string to lower and upper case first, prior to performing the comparison. GENERIC_BASELETTER sort solves these issues but in itself is restrictive, in that it is a general sort not designed for a specific language or locale. The new case and accent insensitive searching is the best of all worlds as it can be applied to any linguistic sort, it won't degrade performance and it allows customers to have the same SQL behaviors without changing existing code.

Specifying a Case-Insensitive or Accent-Insensitive Sort

To specify a case-insensitive or accent-insensitive sort:

- Append _CI to an Oracle sort name for a case-insensitive sort.
- Append _AI to an Oracle sort name for an accent-insensitive and case-insensitive sort.

Examples:

```
BINARY_CI   /*Accent sensitive and case insensitive binary sort */
BINARY_AI   /* Accent insensitive and case insensitive binary sort */
FRENCH_M_AI  /* Accent insensitive and case insensitive 'FRENCH_M' sort */
GENERIC_M_CI   /* Accent sensitive and case insensitive 'GENERIC_M' sort */
```

Example 1. Case and Accent Insensitive Binary Sort

Let's look at a column LETTER with the following data.

```
LETTER
------
ä
a
A
Z
```

The following table shows the sort orders that result from setting the sort to BINARY, BINARY_CI, and BINARY_AI.

1) BINARY	2) BINARY_CI	3) BINARY_AI
A	a	ä
Z	A	a
a	Z	A
ä	ä	Z

1. Compare the results of running a binary sort where, uppercase letters come before lowercase letters. Letters with diacritics appear last.
2. When the sort considers diacritics but ignores case (BINARY_CI), the letters with diacritics appear last.
3. When both case and diacritics are ignored (BINARY_AI), ä is sorted with the other characters whose base letter is a. All the characters whose base letter is a occur before z.

Example 2. Case and Accent Insensitive German SortLet's look again at the column LETTER with the same data but with the sort set to GERMAN.

The following table shows the sort orders that result from from setting the sort to GERMAN, GERMAN_CI, and GERMAN_AI.

GERMAN	GERMAN_CI	GERMAN_AI
a	a	ä
A	A	a

ä	ä	A
Z	Z	Z

1. A German sort places lowercase letters before uppercase letters, and ä occurs before Z.
2. When the sort ignores both case and diacritics (GERMAN_AI), ä appears with the other characters whose base letter is a.

Linguistic Sort Parameters

NLS parameters are used to determine the locale-specific behavior in SQL queries. Most NLS parameters can be configured at the database session level. Switching cultural conventions in a database session is vital for applications that support multiple languages; it allows users with different locale requirements to connect to the same single database.

The parameter NLS_SORT governs the linguistic sort property of the user's SQL session.

Parameter type String
Syntax NLS_SORT = {BINARY | linguistic sort}
Default value Derived from NLS_LANGUAGE
Parameter scope Initialization Parameter, Environment Variable and ALTER SESSION
Range of values BINARY or any valid linguistic sort definition name

NLS_SORT is implicitly defined by another parameter called NLS_LANGUAGE. The value of NLS_SORT may change if the value of NLS_LANGUAGE is altered within a given user session.

The parameter NLS_SORT specifies the collating sequence for ORDER BY queries. If the value is BINARY, then the collating sequence is based on the numeric code of the characters in the underlying encoding scheme. Depending on the datatype, this will either be in the binary sequence order of the database character set or the national character set.

If the value is a named linguistic sort, sorting is based on the order of the defined sort. Most (but not all) languages supported by the NLS_LANGUAGE parameter also support a linguistic sort with the same name.

The table below lists the default NLS_SORT value for each of the Oracle languages. New additions for Oracle Database 10*g* are in bold.

NLS_LANGUAGE	NLS_SORT
AMERICAN	BINARY
ARABIC	ARABIC
ASSAMESE	BINARY
AZERBAIJANI	**AZERBAIJANI**
BANGLA	BINARY
BENGALI	BENGALI
BRAZILIAN PORTUGUESE	WEST_EUROPEAN
BULGARIAN	BULGARIAN
CANADIAN FRENCH	CANADIAN FRENCH
CATALAN	CATALAN
CROATIAN	CROATIAN
CZECH	CZECH
DANISH	DANISH
DUTCH	DUTCH
EGYPTIAN	ARABIC
ENGLISH	BINARY
ESTONIAN	ESTONIAN
FINNISH	FINNISH
FRENCH	FRENCH
GERMAN	GERMAN
GERMAN DIN	GERMAN
GREEK	GREEK
GUJARATI	BINARY
HEBREW	HEBREW
HINDI	BINARY
HUNGARIAN	HUNGARIAN
ICELANDIC	ICELANDIC
INDONESIAN	INDONESIAN
ITALIAN	WEST_EUROPEAN
JAPANESE	BINARY
KANNADA	BINARY
KOREAN	BINARY
LATIN AMERICAN SPANISH	SPANISH
LATVIAN	LATVIAN
LITHUANIAN	LITHUANIAN
MALAY	MALAY
MALAYALAM	BINARY
MARATHI	BINARY
MEXICAN SPANISH	WEST_EUROPEAN
NORWEGIAN	NORWEGIAN
ORIYA	BINARY
POLISH	POLISH
PORTUGUESE	WEST_EUROPEAN
PUNJABI	BINARY
ROMANIAN	ROMANIAN
RUSSIAN	RUSSIAN
SIMPLIFIED CHINESE	BINARY
SLOVAK	SLOVAK
SLOVENIAN	SLOVENIAN
SPANISH	SPANISH
SWEDISH	SWEDISH
TAMIL	BINARY
TELUGU	BINARY
THAI	THAI_DICTIONARY
TRADITIONAL CHINESE	BINARY
TURKISH	TURKISH
UKRAINIAN	UKRAINIAN
VIETNAMESE	VIETNAMESE

In general, a binary sort requires less system overhead; this is because standard Oracle indexes built according to the binary order of the keys are smaller than the linguistic indexes described in the section "Using Linguistic Index".

The NLS_LANG environment variable can also influences the NLS_SORT behavior. NLS_SORT will be changed to the default value assigned to a given NLS_LANGUAGE parameter, as defined by the <language> component of the NLS_LANG environment variable. It is recommended for users to set the NLS_SORT parameter explicitly, to ensure the correct linguistic sequence is being used for a given session.

The examples below illustrate the differences between a binary sort, a monolingual Swedish linguistic sort and a multilingual GENERIC_M linguistic sort.

Example 3. Binary Sort

```
ALTER SESSION SET NLS_SORT=BINARY;

SELECT product_name
FROM   product
ORDER BY product_name;


PRODUCT NAME
------------
Antenne
Lcd
aerial
Ähre
ächzen
```

Example 4. Monolingual Swedish Sort

```
ALTER SESSION SET NLS_SORT=SWEDISH;

SELECT product_name
FROM   product
ORDER BY product_name;


PRODUCT NAME
------------
aerial
Antenne
Lcd
ächzen
Ähre
```

Example 5. Multilingual GENERIC_M Sort

```
SELECT product_name
FROM   product
ORDER BY NLSSORT(product_name, 'NLS_SORT=GENERIC_M');
```

```
PRODUCT NAME
------------
ächzen
aerial
Ähre
Antenne
Lcd
```

When using comparison operators, characters are compared according to their binary codes in the designated encoding scheme. A character is greater than another if it has a higher binary code. Since the binary sequence of characters may not match the linguistic sequence for a particular language, such comparisons may not be 'linguistically correct'. The SQL NLSSORT function allows such comparisons to reflect linguistic conventions.

Example 6. BINARY comparison

```
ALTER SESSION SET NLS_SORT=GERMAN;

SELECT product_name
FROM product
WHERE product_name > 'Antenne'
ORDER BY product_name;

PRODUCT NAME
------------
ächzen
aerial
Ähre
Lcd
```

Example 7. GERMAN linguistic sensitive comparison

The SQL NLSSORT function has to be added on both sides of the comparison operator.

```
ALTER SESSION SET NLS_SORT=GERMAN;

SELECT product_name
FROM product
WHERE NLSSORT(product_name) >
      NLSSORT('Antenne')
ORDER BY product_name;

PRODUCT NAME
------------
Lcd
```

Using NLSSORT function in SQL statements can be cumbersome. Alternatively, you can set NLS_COMP to indicate that the comparison must be linguistically sensitive according to the NLS_SORT session parameter.

Parameter type String
Syntax NLS_COMP = {BINARY | LINGUISTIC | ANSI}
Default value BINARY
Parameter scope Initialization Parameter, Environment Variable and ALTER SESSION
Range of values BINARY, LINGUISTIC, or ANSI

e.g. ***Example 7 NLS_COMP. GERMAN linguistic sensitive comparison*** above can be rewritten using

```
ALTER SESSION SET NLS_SORT=GERMAN;
ALTER SESSION SET NLS_COMP=LINGUISTIC;

SELECT product_name
FROM product
WHERE product_name > 'Antenne'
ORDER BY product_name;

PRODUCT NAME
------------
Lcd
```

Example 8. Multilingual Case insensitive GENERIC_M search

```
ALTER SESSION SET NLS_SORT=GENERIC_M_CI NLS_COMP=LINGUISTIC;

SELECT product_name
FROM product
WHERE product_name LIKE 'A%';

PRODUCT NAME
------------
ächzen
aerial
Ähre
Antenne
```

Note: The NLS_COMP=ANSI setting is available for backward compatibility with pre-10g Release 2 RDBMS only. ANSI supports NLS_SORT comparisons for the following SQL operations: `WHERE, ORDER BY, START WITH, IN/NOT IN, BETWEEN, CASE WHEN,` and `HAVING.` All other SQL operators will compare in binary mode only. To enable linguistic sensitive comparisons across all SQL operations, `NLS_COMP` must be set to LINGUISTIC.

Using a Linguistic Index

Linguistic sorting is language specific and requires more data processing than binary sorting. Binary sorting is fast because it is in the order of the character set encoding. When data of multiple languages are stored in the database, you may want your applications to collate a result set returned from a SELECT statement, using the ORDER BY clause, with different linguistic sequences based upon the language being used, and without the performance penalty associated with linguistic sorting. This can be accomplished by using linguistic indexes. Linguistic index is a variation of the function-based index.

There are three approaches to set up linguistic indexes to sort your language data.

1. Build a linguistic index for each language that the application needs to support. This approach offers simplicity but requires more disk space. For each index, the rows in the language other than the one on which the index is built are collated together at the end of the sequence. The following example builds linguistic indexes for sorting French and German data.

```
CREATE INDEX french_index ON product
(NLSSORT(product_name, 'NLS_SORT=FRENCH'));

CREATE INDEX german_index ON product
(NLSSORT(product_name, 'NLS_SORT=GERMAN'));
```

The index is selected by the SQL optimizer based on the NLS_SORT session parameter or the arguments of the NLSSORT function that is specified in the ORDER BY clause. For example, if the session variable NLS_SORT is set to FRENCH, french_index will be selected and when it is set to GERMAN, the german_index will be used.

2. Build a single linguistic index for all languages using a multilingual linguistic sort such as GENERIC_M or FRENCH_M. This index collates characters according to the character rules defined in the ISO 14651 standard.

```
CREATE INDEX generic_index on product
(NLSSORT(product_name, 'NLS_SORT=GENERIC_M');
```

The index is automatically picked up if the NLS_SORT session parameter or the argument of the NLSSORT function you specified in the ORDER BY clause is equal to the sort name used in the index definition. This approach is useful if the languages that you need to support are covered by a given multilingual linguistic sort.

3. Build a single linguistic index for all languages. This can be accomplished by including a language column in your table (such as LANG_COL in the example SQL below) to be used as a parameter of the NLSSORT function. The language column contains the NLS_SORT values for the data in the column on which the index is built. The following example builds a single linguistic index for multiple languages. With this index, the rows with the same value for NLS_SORT are collated correctly relative to each other. Rows with different values cannot be meaningfully compared.

When creating an index, the total length of the index key cannot exceed a certain value. This value depends primarily on the DB_BLOCK_SIZE. If an attempt is made to create an index with a key larger than the maximum value, an "ORA-1450 maximum key length exceeded" error is raised.

The maximum allowable length of the index key for 2K block is 758, 4K block is 1578, 8K block is 3218 and 16K block is 6498.

```
CREATE INDEX nls_index ON product
(NLSSORT(product_name, 'NLS_SORT=' || LANG_COL));
```

The nls_index will be used only if the query explicitly specifies `NLSSORT (product_name, 'NLS_SORT=' || LANG_COL)` in the `ORDER BY` clause.

As with other function-based indexes, building composite linguistic indexes are also supported.

For example:

```
CREATE INDEX german_index ON product
(NLSSORT(product_name, 'NLS_SORT=GERMAN'),
 NLSSORT(company_name, 'NLS_SORT=GERMAN'));
```

In fact, the rule based optimizer can use the non-functional prefix of a composite linguistic index, if there is one, i.e. if the above index was modified to include the model number,

```
CREATE INDEX german_index ON product
(model_number, NLSSORT(product_name, 'NLS_SORT=GERMAN'),
 NLSSORT(company_name, 'NLS_SORT=GERMAN'));
```

then a rule-based query can take advantage of this composite linguistic index as well.

Requirements for Using Linguistic Indexes

Whether you decide to use a single linguistic index or multiple linguistic indexes, two requirements must be met for the linguistic index to be utilized.

1. Set NLS_SORT appropriately. The first requirement is that the NLS_SORT parameter for the query should indicate the linguistic definition that was specified in the index CREATE statement.
2. Specify NOT NULL in WHERE clause if the column was Not Declared NOT NULL. When you want to use the ORDER BY column_name clause with a column that has a linguistic index, include a WHERE clause like the following:

   ```
   WHERE NLSSORT(column_name) IS NOT NULL
   ```

   This WHERE clause is not necessary if the column has already been defined as a NOT NULL column in the schema.

Here is an example on how to create a linguistic index called NLS_GENERIC based on the multilingual linguistic sort GENERIC_M on the PRODUCT table.

```
CREATE INDEX NLS_GENERIC ON product
(NLSSORT(product_name, 'NLS_SORT=GENERIC_M'));

ALTER SESSION SET NLS_SORT=GENERIC_M;

SELECT * FROM product
WHERE NLSSORT(product_name) IS NOT NULL
ORDER BY product_name;
```

Even if all the linguistic index requirements are met, it is possible that the optimizer will choose not to use the linguistic index because there are cheaper plans available.

Adding the hint /*+ index(table indexname) */ will cause the cost based optimizer to be used and will cause the index to be used if there is any legal query plan that allows it

The WHERE clause is not needed if column product_name has been defined as a NOT NULL column.

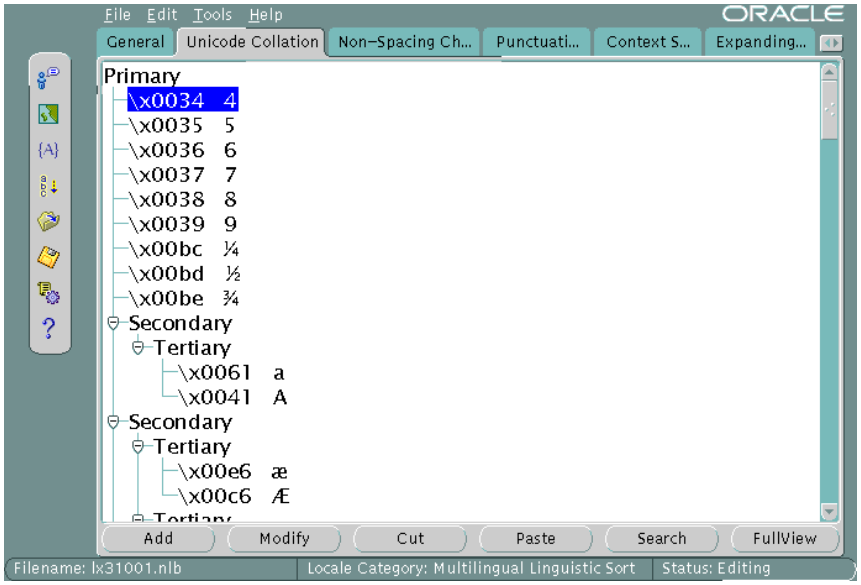
Customization of Linguistic Sorts

A comprehensive collection of linguistic sorts has been provided to meet the demand of our customers with increasing different language needs. However, there will always be new sorting requirements, due to changes in cultural conversions or the result of emerging new industry standards (such as the ISO and Unicode standard). Sometimes it may be necessary to customize a linguistic sort in a way such that it is consistent with the approach adopted by other vendors in order to be compatible with other software products across different platforms.

Oracle Locale Builder is a GUI tool for configuring locale data definitions. It provides an easy-to-use graphical interface through which user can easily view existing Oracle locale definitions, customize them, or create new definitions. For linguistic sort definitions, Locale Builder gives a graphical representation of the sort order that is intuitive for both viewing and customization purposes. Oracle Locale Builder supports the rearrangement of characters in different sequence orders, and

provides customization of contracting and expanding characters, context sensitive characters as well as supplementary characters.

Screen shot of the Oracle Locale Builder.



Please refer to *Oracle Database Globalization Support Guide* for more information.

SUMMARY

Sorting your data in a linguistic sensitive manner is an important part of data processing in a globally deployed application. Presenting data not sorted in the linguistic sequence that your users are accustomed to can make searching for information difficult and time consuming.

Oracle has expanded the coverage of binary sorts, linguistic sorts and introduced the new case & accent insensitive sorts to meet the demands of customers who need to search and sort data across the different languages. Oracle continues to enhance linguistic sort coverage and provides conformance to international standards by supporting the ISO 14651 - International string ordering standard and the Unicode collation standard (UCA).

Oracle offers a comprehensive selection of linguistic sorts, supporting over 60 monolingual linguistic sorts and 13 multilingual linguistic sorts. Those customers with special requirements that go beyond the extensive set of linguistic sorts provided by Oracle, have the flexibility of customizing and defining their own

linguistic sorts by using the Oracle Locale Builder, a new easy-to-use GUI tool that allows them to view existing and create new linguistic sorts.

Sorting your data inside the Oracle database
May 2005
Author: Simon Law
Contributing Authors: Sergiusz Wolicki, Claire Ho, Barry Trute

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com