WG3: BHX-080

08 June 2000

Authoritative version: bhx080.pdf

ISO

International Organization for Standardization

**ISO/IEC JTC 1/SC 32 Data Management and Interchange WG3 Database Languages**

**Secretariat: USA (ANSI)**

| | |
|---|---|
| **Project:** | 32.03.05.02.00 |
| **Title:** | A Review of Some Possible Problems with SQL character features |
| **Author:** | J M Sykes (United Kingdom) |
| **Source:** | UK Expert |
| **Status:** | Discussion paper |

**Abstract:** A study of SQL Features F451, "Character set definition", F461, "Named character sets" and F691, "Collation and translation" has revealed several problems which cannot be resolved without discussion.

**References:**

| | |
|---|---|
| [Framework-99] | ISO/IEC 9075-2:1999, Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework) |
| [Found-99] | ISO/IEC 9075-2:1999, Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation). |
| [FrameworkWD] | *(ISO Working Draft) Database Language SQL - Part 1*: *Framework (SQL/Framework*), March, 2000 |
| [FoundWD] | *(ISO Working Draft) Database Language SQL - Part 2: Foundation (SQL/Foundation*), March, 2000 |

[SchemataWD] *(ISO Working Draft) Database Language SQL - Part 10: Schemata (SQL/Schemata)*, March, 2000

[MAD-145] SQL standard character set names, L Gallagher, November 1996

[CWB-051] Character set internationalization revisited, J Melton, January 1998. (Note: this paper contains references to ten papers on the subject considered by the Florence meeting of ISO DBL in October, 1989)

[BHX-049] On Three Different Kinds of Sameness, JM Sykes, April 2000, (WG3: BHX-049)

[BHX-053] A Compendium of Possible Problems in SQL: 1999, Hugh Darwen (Ed.), May 2000 (WG3:BHX-053)

[ISO10646] ISO/IEC 10646-1:2000, Information technology — Universal Multi-Octet Coded Character Set(UCS) — Part 1: Architecture and Multilingual Plane

[ISO14651] ISO/IEC DIS 14651 – International string ordering and comparison – Method for comparing character strings and description of the common template tailorable ordering, April 2000

[Unicode] *The Unicode Standard, Version 3.0*, The Unicode Consortium, Feb 2000

[UnicodeGlossary] http://www.unicode.org/glossary

[UnicodeTechIntro] http://www.unicode.org/unicode/standard/principles.html

[UnicodeFAQ] http://www.unicode.org/unicode/faq

[UnicodeTR#10] Unicode Collation Algorithm, (Unicode Technical Report #10), Revision 5.0, November 1999

[UnicodeTR#15] Unicode Normalization Forms (Unicode Technical Report #15), Revision 18.0, November 1999

[UnicodeTR#17] Character Encoding Model (Unicode Technical Report #17), Revision 3.0, November 1999

## 1 Introduction

In the course of drafting [BHX-049], the search for a satisfactory definition of *identical* in the case of character strings revealed some curious inconsistencies in the SQL standard, some of which are editorial and not difficult to correct, while others seem to be more fundamental. Some of these are documented in [BHX-053]. Further investigation has led us to review the whole treatment of character data and to question whether SQL's facilities for defining and manipulating such data are either necessary or sufficient.

The body of this paper consists of an overview of the history of SQL's treament of character data, followed by our view of the present situation and what should be done about it. There follow annexes giving respectively: a review of the views and stated intentions of CWB-051, with our views thereon; a selection of definitions from sources other than SQL; a review of character-related definitions in [Found99]; and an attempt to answer CWB-051's question 'What is a character?' according to Unicode.

# 2 A brief history of characters in SQL

## 2.1 SQL-87 and SQL-89

The only character type was CHARACTER. Neither specified either NATIONAL CHARACTER or VARYING CHARACTER.

## 2.2 SQL-92

SQL-92 introduced data types that had been implemented by some vendors, such as NCHAR and VARCHAR, and some additional features such as concatenation and substring.

It also introduced character sets, collations and translations, mostly in a number of papers processed at the Florence meeting in October 1989. Much, but by no means all, of the resulting language was published in SQL-92, though none in Entry level, and not a lot in Intermediate level. The remainder was carried over into SQL3.

## 2.3 SQL-99

Two important papers were processed before SQL-99 was published:

### 2.3.1 MAD-145

In January 1997, MAD-145 introduced the names of standard character sets, as specified in the U.S. FIPS publication for SQL. These were: SQL_CHARACTER, GRAPHIC_IRV (or ASCII_GRAPHIC), LATIN1 and ISO8BIT (or ASCII_FULL).

### 2.3.2 CWB-051

In January 1998, CWB-051 made a serious attempt to simplify what had become a somewhat untidy and possibly unimplementable specification. The discussion section of CWB-051 contained much factual background material that is still relevant, and we refer the reader particularly to the first three subsections:

1.1 A brief history of SQL-92 character internationalization,

1.2 Motivation for SQL-92 character internationalization, and

1.3 An outline of SQL-92 character internationalization

The following sections are considered in more detail in Annex A.

The proposal in Section 2 of CWB-051 also made some changes that are not mentioned in Section 1. For example, it introduced two subclauses listing a number of terms defined in ISO/IEC 10646 and Unicode respectively, each of which asserts that "This part of ISO/IEC 9075 makes use of the following terms ...", when in fact it uses hardly any of them. Since the proposal was large and the meeting that processed it had a heavy agenda, it would not be surprising if there were more such cases.

Clearly, CWB-051 set out to be a bold step in the right direction, but it failed to deliver on some of its promises, and, in our view, was not bold enough.

## 2.4 Since SQL-99

### 2.4.1 BHX-049 On Three Kinds of Sameness

This paper needed to specify how to determine whether two character values were identical. The first approach to be considered, and actually adopted in the first version, was based on comparing the results of casting the two character strings to bit strings, as being simpler than getting involved with collations and other details of Features 451, 461 and 491. Following further discussion, it was decided to discard this approach and face up to collations. However, the authors became increasingly concerned with the specification of these features, and found a number of problems, varying in severity from minor editorial to apparently more severe ('apparently' because of lack of clarity). It was decided to document these problems as formal comments, and to include all that were found in time in the UK's comments on the SQL DCOR. However, because they all relate to problems in [Found99], we have not assumed that any consequent changes to the SQL specification will necessarily be reflected in the Corrigendum when published. So, to make this expectation a little more clear, the title of [BHX-053] was changed.

### 2.4.2 BHX-053 A Compendium of Possible Problems in SQL: 1999

Of the 43 comments listed, about half have been generated as a result of our researching aspects of character sets. More have been found since.

# 3 Overview of the present situation

## 3.1 Summary of the facts

o As we have noted above, CWB-051, having reviewed the situation as it was in early 1998, expressed the intention to make certain changes, but didn't do all it intended;

o A number of possible problems have been found in the specification of Features 451, 461 and 691;

o As far as we know, in the eight years since these features have been in the SQL standard (in one form or another), no one has yet claimed to have implemented any of them. However, at least one implementor is understood to have provided support for Unicode (by hiding it behind NATIONAL CHARACTER), while another is known to be uncomfortable with at least certain aspects of collations.

o Acceptance of Unicode has increased in the two years since CWB-051: version 3.0 was published in February 2000, and, to quote the Unicode FAQ Web page:

> The industry is converging on Unicode for all internationalization. For example, Microsoft NT is built on a base of Unicode; AIX, Sun, HP/UX all offer Unicode support. All the new web standards; HTML, XML, etc. are supporting or requiring Unicode. The latest versions of Netscape Navigator and Internet Explorer both support Unicode. **Sybase, Oracle, DB2** all offer or are developing Unicode support. [emphasis added]

o The issue of alternative Unicode representations (mentioned in CWB-051, section 1.9.3 "OK, what is a character?") is now more comprehensively addressed in [Unicode TR15], "Unicode Normalization Forms" and in [UnicodeTR17] "Character Encoding Model".

Moreover, the former includes the statement:

> The Unicode Standard provides a recommended syntax for identifiers for programming languages that allow the use of non-ASCII languages in code. It is a natural extension of the identifier syntax used in C and other programming languages.

- SC22 (Programming languages, their environments and system software interfaces) /WG22 (Internationalization) has recently produced [ISO14651] (ISO/IEC DIS 14651 – "International string ordering and comparison – Method for comparing character strings and description of the common template tailorable ordering"). CWB-051 expressed reservations about this work in section 1.7 "Some caveats"; whether these reservations have been withdrawn we have not yet discovered. However, [UnicodeTR10] " Unicode Collation Algorithm" also exists, and includes the statement:

  > The Unicode Collation Algorithm assumes multiple-level key weighting, ... as described in ... the proposed International String Ordering standard (ISO/IEC 14651).

## 3.2 Proposed action

- Features 451, 461 and 691 should be notionally removed from the SQL standard;
- The situation should then be reviewed, much as was done by CWB-051, to determine the requirements for an SQL-implementation to be able to deal with:
  - character data encoded in more than one way. In this review, both SQL/MED and SQL/OLB must be considered, as well as data transfer to and from host languages;
  - identifiers containing characters not in the repertoire of <sql language character>.
- If and only if necessary (bearing in mind that NCHAR &c. are already there) an SQL feature should be specified to satisfy those, and only those, requirements.

## 3.3 Guidelines

- In whatever is specified, terms defined in other standards (including The Unicode Standard) shall be used with the meanings defined in those other standards, except in the case that the definition in another standard is inadequate.
- Specifications from other standards shall be referenced wherever possible.
- Text from other standards shall not be repeated in the SQL-standard.
- Nothing should be specified in the SQL standard that is or ought to be in some other standard.
- At least informal liaison should be established with: SC 2, SC 22/WG20 and the Unicode Consortium.
- What implementors have already done or are planning to do must be considered.

## 3.4 Issues needing to be addressed

### 3.4.1 How many named character sets?

- Which does the SQL database designer need to be able to specify, out of: UTF8, UTF16, UTF16LE, UTF16BE, UTF32, UTF32LE, UTF32BE?

  (Note. [CWB-051] proposed to insert a paragraph into SQL/Foundation that said:

  – **UNICODE** and ISO10646 specify the name of a character repertoire that consists of every character represented by The Unicode Standard Version 2.0 and ISO/IEC 10646 UTF-16, where each character is encoded using **exactly 16 bits.**

  In [Found99] this has been amended to:

- **UTF16** and ISO10646 specify the name of a character repertoire that consists of every character represented by The Unicode Standard Version 2.0 and by ISO/IEC 10646 UTF-16, where each character is encoded using the **UTF-16 encoding, occupying either 1 (one) or 2 octets.**

The statement appears still to be incorrect, because [UTR#17], section 4 "Character Encoding Form (CEF)", says:

o UTF-16 [is] used only with Unicode/10646: a mix of **one to two 16 bit code units**.

o Where and how should the syntax allow such character set names to be specified?

### 3.4.2 Normalization

From [UnicodeTR#5]:

**Note:** Text containing only ASCII characters (U+0000 to U+007F) is left unaffected by all of the normalization forms. This is particularly important for programming languages.

Sigh of relief for SQL code!

o There are four forms of normalization. How many should SQL recognise? Which should they be?

o What data can an SQL-implementation ever, sometimes or never assume to be normalized? SQL-data, character literals, character values passed as parameters or received as results? If unnormalized data is found, should an exception be raised, or the data be normalized forthwith, or only if necessary?

o Should 'normalized' be specifiable as a constraint?

o Should a NORMALIZE function be specified?

o What should be done about concatenation, in view of "None of the normalization forms are closed under string concatenation." ([UnicodeTR#5], Introduction)

### 3.4.2 Collation

o How should collations be made available?

o How shall the collation to be used be specified, taking into account current implementations.

### 3.4.3 Counting characters

o How should characters (particularly composite characters) be counted, for the purposes of

- the results of CHAR_LENGTH, POSITION and OVERLAY
- the parameters of SUBSTRING, OVERLAY, LIKE and SIMILAR?

### 3.4.4 Parameter and result passing

o How should Unicode strings be passed as parameters to (or received as results from) host languages that are not 'Unicode-aware'?

## 3.5 Possible outcomes (not exclusive)

o An implementor might allow a user to specify that NCHAR is to mean Unicode (or, presumably, something else).

- A new data type, UCHAR, might be introduced, meaning Unicode.
- A means might be provided to specify a constraint that a Unicode character column contain only some subset of Unicode, for example Sanskrit or Cyrillic.
- Names might be defined for standard functions to convert from one character set to another, for example CHAR_TO_NCHAR(), ASCII_TO_EBCDIC().
- Names might be defined for standard collations, for example ISO8BIT_CASEBLIND, and a method provided for specifying their use, possibly at the time of their use only.
- A means might be provided of designating a (user-defined) function as a collation, in much the same way as a udt ordering function.

# Annex A: Detailed review of CWB-051

Note: Heading numbers are taken from [CWB051]

| ***ORIGINAL TEXT*** | **Our comment** |
|---|---|
| 1.4 What's right and what's wrong with SQL-92 character internationalization | |
| In spite of the problems and complexity of SQL-92's character internationalization facilities, some things were done in a reasonable way; these things will require few (if any) changes. Other things were done in ways that have proved to be naïve, unusable, or unpopular; these will be changed significantly or eliminated entirely. | To the 'Other things', we would add 'unimplementable'. Sadly, we believe not enough was changed or removed. |
| **1.4.1 What's right** | |
| • The ability to specify a character set for a given column in a table is needed and should be retained. | Indisputable. |
| • The ability to specify different character sets for different columns in a table is required and should be retained. | Indisputable. |
| • The ability to specify a default character set for all columns in all tables in a single schema is clearly appropriate and should be retained. | Debatable, but not objectionable. |
| • The ability to express the names of schemas, tables, column, and other objects in the character set of a user's choice is required and should be retained. | Debatable, and removed by the time ISO saw this paper. |
| • The ability to specify a default collation for a column, parameter, or other character string object that is different from the default collation of the object's character set is needed and should be retained. | Debatable. The need for this is not clear. |
| • The rules for determining the collation to be used for a given operation or applied to the result of an operation are demonstrably correct and should be retained. | Debatable. It would suffice to specify a collation only on an operation. |
| • The need for a "union" character set (SQL_TEXT) that represents all characters in all supported character sets is obvious and should be retained for use in the Information Schema and in various descriptors. | Probably. |
| **1.4.2 What's wrong** | |
| • The attempt to allow an application writer to specify a (different) character set for a specific identifier or character string literal has been widely derided and is obviously unsuccessful; it should be eliminated in favor of requiring a single — but specifiable — character set to be used for any single entity such as an SQL-client module or "standalone" schema definition. | Indisputable, except that the proposal to retain the *module character set* could be debated. |

| ***ORIGINAL TEXT*** | **Our comment** |
|---|---|
| • The ability to define new character sets using SQL language is clearly over-kill; furthermore, it is no longer desirable or necessary, since there are widely-accepted conventions (if not actual *de jure* standards) for associating programming language identifiers with character sets. | Indisputable. |
| • The ability to define collations using SQL language is clearly over-kill; furthermore, it is no longer desirable or necessary, since there is now a *de jure* standard for defining custom collations and there are at least a small number of *de jure* standards for collations. | Indisputable. |
| • The ability to define translations using SQL language is clearly over-kill; regrettably, there is nothing appropriate to replace it, so solutions are likely to be awkward at best. | Indisputable that there was over-kill, and something is probably needed. |
| • The definition of SQL_TEXT is unnecessarily cumbersome; its implementation-defined characteristic doesn't help application portability. It is more appropriate to select a specific character set — for which the present paper will select UCS/UTF-16 (a/k/a Unicode) — for this purpose. | We are sympathetic to this; but it's not clear that the proposal achieves the stated objective. |
| • There is not a unifying principle for character sets; specification of such a principle would greatly simplify efforts to explain what SQL supports in this area. | We have not noticed such a *unifying principle*. |
| • The alignment of SQL with host languages is inadequate and should be enhanced. | Indisputable. |

## 1.5 Motivation for the approach taken by this proposal

| | |
|---|---|
| The approach taken by the present proposal is driven by several factors: | |
| • The commercial influence and success of Unicode (used, for example, by HTML, Java, and Windows NT), along with the congruence between that specification and the international standard ISO/IEC 10646. | Indisputable |
| • The (fairly) recent publication of an international standard that provides the ability to define (customize) collations for any language and any script, along with the publication of a few national standards (at least Canada and Germany — possible more) for specific collations. | Indisputable |
| • Firm, even resolute, input from database system vendors that the SQL-92 character internationalization specifications are too complex and not nearly relevant enough for their products and customers. | Indisputable |

## 1.6 Outline of the present proposal

| | |
|---|---|
| The present proposal offers changes to several parts of SQL3 related to character internationalization. Of course, the most pressing changes are those proposed to the Final Committee Draft documents, for which the present proposal offers solutions to one or more National Body comments on the FCD ballots for those documents. In the spirit of keeping the future aligned with the | No comment called for. |

| ***ORIGINAL TEXT*** | **Our comment** |
|---|---|
| present, corresponding changes are offered for the Working Draft documents of the same parts for which FCD ballots were taken; similarly, changes are offered for the WD documents of any other parts that are affected by the subject being proposed. | |
| The changes can be summarized thus: | |
| • In SQL/Framework, modify and significantly enhance the Concepts related to character internationalization — both aligning with the new specifications proposed for other parts and increasing the coverage and understandability of the presentation. | While we agree *something* has to go in SQL/Framework, we seem to have achieved fragmentation, possibly with some undesirable duplication. |
| • In SQL/Foundation: | |
| – Eliminate the ability to prefix identifiers with <underscore><character set specification> | Clearly done. |
| – Eliminate the ability to define new character sets “from scratch”, but retain the ability to define a new character set identical to another character set (thus providing compatibility for applications already using one character set name in spite of standards assigning a different identifier to the same character set); do the same for collations. For translations, make the existing description in SQL/Framework accurate by restricting its capabilities to either specification of a named translation or of an SQL-invoked function used to perform the translation. | Done, but the value of having what is essentially only an aliasing facility of debatable value.<br><br>The ability to designate an SQL-invoked function as a translation might be useful. |
| – Add ISO/IEC 10646 UTF-16 (a/k/a Unicode) as a defined character set to the list already in SQL/Foundation. | Done, but the definitions are not sufficiently precise. |
| – Ensure that Unicode is the “canonical form” for specifying character sets (that is, every character in every character set must be “mappable” to a Unicode character) without including language requiring that Unicode actually be mandated for user data or for metadata other than in the Information Schema. (I have looked to HTML for guidance here.) Note that Unicode accomodates “private use characters”, which allow implementation-defined character sets containing characters not published in the Unicode standard to be supported without violating this requirement. | Reasonable, but needs discussion. The use of quotes suggests some lack of precision. |
| – Put into place SQL specifications (syntax where appropriate, rules elsewhere) supporting the referencing of registries of character sets, etc., for use in specifying identifiers for character sets, etc. | Reasonable. But we have been unable to discover how this is achieved. |
| – Modify the specification of identifiers to use Unicode-like language to describe permitted characters. | If *Unicode-like language* means *Unicode notation,* then: yes. |

| ***ORIGINAL TEXT*** | **Our comment** |
|---|---|
| – Modify the Information Schema views (and the corresponding Definition Schema tables) related to character sets, collations, and translations to match the new specifications. | Done. |
| • In SQL/PSM: | |
| – Make minor changes to keep aligned with SQL/Foundation. | |
| • In SQL/Bindings: | |
| – Make minor changes to keep aligned with SQL/Foundation. | |
| – Make changes to the host language bindings to take advantage of the new character model. | The changes were "To be supplied", and appear not to have been. |

# Annex B: ISO/IEC 10646 and Unicode definitions

## B.1 Source text

Study of the Unicode Web pages (on the web site) has narrowed this issue down to the following (lightly edited) definitions taken from the Glossary, and presented here in logical, rather than alphabetical, order, where each term is defined as late as possible before use.

*Italic* is original, **bold** has been added in the hope of helping the reader.

| ***From Unicode Glossary*** | ***From Unicode Technical Introduction (except <JMS> and <UTR#17>)*** |
|---|---|
| ***Character***. (1) The smallest component of written language that has semantic value; refers to the abstract meaning and/or shape, rather than a specific shape (see also glyph), though in code tables some form of visual representation is essential for the reader's understanding. (2) Synonym for abstract character. (3) The basic unit of encoding for the Unicode character encoding. | <JMS> (1) and (2) are consistent with each other, and with the view that characters are analogous to numbers, in being abstract, and that EBCDIC and ASCII, as representations of characters, are analogus to binary and decimal, or half and full words, as representations of numbers. |
| ***Character Set***. A collection of elements used to represent textual information. | |
| *Character Repertoire*. The collection of characters included in a character set. | <UTR#17> ***Abstract Character Repertoire* (*ACR*).** The set of characters to be encoded, e.g., some alphabet or symbol set |
| ***UCS***. Abbreviation for Universal Character Set, which is specified by International Standard ISO/IEC 10646. | |
| *Code Element*: (no definition in Glossary) | To avoid deciding what is and is not a text element in different processes, the Unicode Standard defines *code elements* (commonly called "characters"). A code element is fundamental and useful for computer text processing. |
| *Code Value*. The minimal bit combination that can represent a unit of encoded text for processing or interchange. | A single 16-bit number is assigned to each code element defined by the Unicode Standard. Each of these 16-bit numbers is called a *code value* and, when referred to in text, is listed in hexadecimal form following the prefix "U". |
| *Coded Character Set*. A character set in which each character is assigned a numeric code value. Frequently abbreviated as character set, charset, or code set. | ***Coded Character Set* (*CCS*)**. a mapping from an abstract character repertoire to a set of non-negative integers |

| ***From Unicode Glossary*** | ***From Unicode Technical Introduction (except <JMS> and <UTR#17>)*** |
|---|---|
| (*Code Page*. A coded character set, often referring to a coded character set used by a personal computer--for example, PC code page 437, the default coded character set used by the U.S. English version of the DOS operating system.) | |
| *Form of use*: (no definition in Glossary) | The international standard ISO/IEC 10646 allows for two **forms of use**, a two-octet (=byte) form known as **UCS-2** and a four-octet form known as UCS-4. The Unicode Standard, **as a profile of** ISO/IEC 10646, **chooses** the two-octet form, which is equivalent to saying that characters are represented in 16-bits per character. When extended characters are used, **Unicode is equivalent to UTF-16**. |
| (*Character*) ***Encoding Form***. Mapping from a character set definition to the actual bits used to represent the data. | **<UTR#17> *Character Encoding Form (CEF).*** A mapping from a set of non-negative integers (from a CCS) to a set of sequences of particular code units of some specified width, such as bytes<br><br><JMS> Unicode *Character Encoding Form* appears to be synonymous with ISO *form of use*. |
| ***UCS-2***. ISO/IEC 10646 encoding form (form of use): Universal Character Set coded in 2 octets. | |
| (*UCS-4*. ISO/IEC 10646 encoding form: Universal Character Set coded in 4 octets.) | |
| *Abstract Character*. A unit of information used for the organization, control, or representation of textual data. | |
| *Coded Character Representation*. An ordered sequence of one or more code values that is associated with an abstract character in a given character repertoire. | |
| *Coded Character Sequence*. An ordered sequence of coded character representations. | |
| *Transformation Format*. A mapping from a coded character sequence to a unique sequence of code values (typically bytes). | The Unicode Standard **endorses** two forms that correspond to ISO 10646 *transformation formats*, UTF-8 and UTF-16.<br><br>The ISO/IEC 10646 transformation formats UTF-8 and UTF-16 are essentially ways of turning the encoding into the actual bits that are used in implementation. |

| ***From Unicode Glossary*** | ***From Unicode Technical Introduction (except <JMS> and <UTR#17>)*** |
|---|---|
| | <UTR#17> ***Character Encoding Scheme*** **(*CES*)**. a mapping from a set of sequences of codes units (from one or more CEFs) to a serialized sequence of bytes. |
| ***UTF-8***. Unicode (or UCS) Transformation Format, 8-bit encoding form. UTF-8 is the Unicode Transformation Format that serializes a Unicode scalar value as a sequence of one to four bytes, ... | **UTF-8** is a way of transforming all Unicode characters into a variable length encoding of bytes. It has the advantages that the Unicode characters corresponding to the familiar ASCII set end up having the same byte values as ASCII, and that Unicode characters transformed into UTF-8 can be used with much existing software without extensive software rewrites. |
| Big/little-endian. A computer architecture that stores multiple-byte numerical values with the most/least significant byte (MSB/LSB) values first. (It's not clear how important this is, being a rather low level hardware issue.) | |
| ***UTF-16***. Unicode (or UCS) Transformation Format, 16-bit encoding form. The UTF-16 is the Unicode Transformation Format that serializes a Unicode value as a sequence of two bytes, in either big-endian or little-endian format. | **UTF-16** assumes 16-bit characters and allows for a certain range of characters to be used as an extension mechanism in order to access an additional million characters using 16-bit character pairs. *The Unicode Standard, Version 2.0*, has adopted this transformation format as defined in ISO/IEC 10646.<br><br>**Any Unicode character expressed in the 16-bit UTF-16 form can be converted to the UTF-8 form and back without loss of information.**<br><br>The Unicode standard and ISO 10646 provide an extension mechanism called UTF-16 that allows for encoding as many as a million more characters, without use of escape codes. |
| ***UTF-16BE/LE***. The Unicode Transformation Format that serializes a Unicode value as a sequence of two bytes, in big/little-endian format. ... | |
| *Collation*. The process of ordering units of textual information. Collation is usually specific to a particular language. Also known as *alphabetizing* or *alphabetic sorting*. Unicode Technical Report #10, "Unicode Collation Algorithm," defines a complete, unambiguous, specified ordering for all characters in the Unicode Standard. | <JMS> Defined in [ISO14651] as:<br><br>equivalent to the term "ordering",<br><br>which in turn is defined, as one would expect, as:<br><br>a process by which two strings are determined to be in exactly one of the relationships of **less than**, **greater than**, or **equal** to one another |

# Annex C: Review of character-related definitions in [Found-99]

## C.1 Subclause 3.1.1 "Definitions taken from ISO/IEC 10646"

(BHX-053, Comment GBR-STC-006)

This subclause states that "this part of ISO/IEC 9075 makes use of" six terms, but in actual fact uses only two, *character*, and *repertoire*. However, it redefines the latter, without explicitly stating whether it is unnecessarily repeating the definition in ISO/IEC 10646, or replacing it.

It (Part 2, Foundation) does not make use of the terms *coded character*, *coded character set*, *control function*, or *private use plane*.

## C.2 Subclause 3.1.2 "Definitions taken from Unicode"

(BHX-053, Comment GBR-STC-007)

This subclause states that "this part of ISO/IEC 9075 makes use of" six terms, but uses only one: *control character* (in the context of ISO8BIT); another, *code value*, occurs only in Clause 22.1 "SQLSTATE", Table 27-"SQLSTATE class and subclass values".

## C.3 Subclause 3.1.5 "Definitions provided in Part 2"

Curiously, but perhaps wisely, no claim is made that the terms defined here are made use of. The following are all the terms relevant to the subject of character sets.

| ***SQL Definition*** | ***Our comment*** |
|---|---|
| *character repertoire*; repertoire: A set of characters used for a specific purpose or application. Each character repertoire has an implied default collating sequence. | Defined externally, with, as far as we can tell, the same meaning, so ought not to be repeated.<br><br>In any case, *A set of characters* looks suspiciously like the term *character set* used elsewhere, the nearest thing to a definition being in SQL/Framework. The word *implied* is either a pleonasm (i.e. unnecessary) or its effect is unclear - could a character repertoire have an *other-than-implicit* default collating sequence? (The only use of *explicit default* that we can find is in connection with columns &c that have one as a result of being defined with a <default clause>.) |
| *coercibility*: A characteristic of a character string value that governs how a collating sequence for the value is determined. | This definition disregards the fact that it takes two values to be collated. In effect, the <collate clause> in, for example, a <column definition> says "When you compare me, please use this collation". Coercibility indicates the degree of insistence in the case where two values have different (preferred ) collations. Hence it is more strictly a characteristic of a declared type. |

| ***SQL Definition*** | ***Our comment*** |
|---|---|
| *collation*; *collating sequence:* A method of ordering two comparable character strings. Every character set has a default collation. | Defined externally, with same meaning.<br><br>One is left wondering whether this is intended to repeat the statement in the definition of *character repertoire*, above, i.e. whether *character set* and *character repertoire* are different terms for the same concept - a view encouraged by the unnecessary variation from *collating sequence* to *collation*. |
| *form-of-use*: A convention (or encoding) for representing characters (in character strings). Some forms-of-use are fixed-length codings and others are variable-length codings. | Defined externally, with essentially the same meaning; but Unicode seems to prefer *transformation format* for this meaning.<br><br>Nothing is said (here) about the relationship of a form-of-use to other concepts. For example, can a form-of-use be associated with more than on character repertoire/set? It looks as though it might be algorithmic, but evidence elsewhere suggests otherwise. |
| *form-of-use conversion*: A method of converting character strings from one form-of-use to another form-of-use. | Looks like what Unicode would call *Transcoding*. (Conversion of character data between different character sets.) |
| *repertoire*: See character repertoire. | Unnecessary. The term is allegedly defined in [ISO/IEC 10646], is defined with *character repertoire*, but in any case is used only a few times. |
| *translation*: A method of translating characters in one character repertoire into characters of the same or a different character repertoire. | Looks more like what the dictionary calls *transliteration*. Why do functions of this class need a special name?<br><br>In any case, it is inconsistent to define a translation as translating **characters**, while a form-of-use conversion converts character **strings.** Subclause 11.34 "<translation definition>", is expressed in terms of character **sets**. |

## C.4 *Character set* and *character repertoire*

[Found99] contains no explicit definition of *character set*. The definition is in [Frame99] Subclause 4.6.2.1 "Character sets", where it says:

> A character set is a named set of characters (character repertoire) that may be used for forming values of the character data type.
>
> ...
>
> This International Standard uses the phrases "character set" and "character repertoire" interchangeably **except when referring to data exchanged outside the SQL-implementation,** when "character set" is understood to include an encoding and a form-of-use in addition to a character repertoire.

This is not clear, and is arguably in the wrong place.

## Annex D: What is a character?

The following definitions are from [UnicodeGlossary]

*Abstract Character*. A unit of information used for the organization, control, or representation of textual data.

*Encoded Character*. An *abstract character* together with its associated *Unicode scalar value*. By itself, an abstract character has no numerical value, but the process of "encoding a character" associates a particular Unicode scalar value with a particular abstract character, thereby resulting in an "encoded character." *Assigned Character* is a synonym for *encoded character*.

*Base Character*. A character that does not graphically combine with preceding characters, and that is neither a control nor a format character.

*Combining Character*. A character that graphically combines with a preceding base character. The combining character is said to *apply* to that base character.

*Coded Character Representation*. An ordered sequence of one or more code values that is associated with an abstract character in a given character repertoire.

*Compatibility Character*. (1) A character encoded only for compatibility with preexisting character encoding standards to support transcoding. (2) A character that has a compatibility decomposition.

*Control Codes*. The 65 characters in the ranges U+0000..U+001F and U+007F..U+009F. Also known as *control characters*.

*Decomposable Character*. A character that is equivalent to a sequence of one or more other characters, according to the decomposition mappings found in the names list of Section 14.1, Character Names List . It may also be known as a *precomposed character* or a *composite character*.

*Formatting Codes*. Characters that are inherently invisible but that have an effect on the surrounding characters.

*Graphic Character*. (1) A character typically associated with a visible display representation. (See also *glyph*.) (2) Any character that is not primarily associated with a control or formatting function.

From the above, one might infer that every Unicode code value is in one of four categories, being one of:

> Base character. A (coded) character that does not graphically combine with preceding characters.
>
> Combining Character. A (coded) character that graphically combines with a preceding base character. Also called 'non-spacing mark'. The combining character is said to apply to that base character.
>
> Control Code. The 65 characters (codes?) in the ranges U+0000..U+001F and U+007F..U+009F. Also known as control character.
>
> Formatting Code. A Character (code?) that is inherently invisible but that has an effect on the surrounding characters.

However, it is difficult to be sure these categories are exhaustive, partly because it is not always clear when 'character' means 'abstract character' and when it means 'coded character', or even 'code value'.

> A composite character is an abstract character that can be represented by a composed character sequence consisting of a base character and one or more combining characters.
>
> A composite character that can be represented by a single base character is called a precomposed character.

> A precomposed character can be decomposed (hence is sometimes called a decomposable character) into a sequence of one or more other characters, according to the decomposition mappings found in the names list of Section 14.1, Character Names List.

It seems to follow that there can be composite (abstract) characters that are not precomposed, and hence cannot be represented by a single base character. There appears to be no term to denote such an abstract character. More importantly, it needs to be decided whether SQL counts each such composite characters as one abstract character, or counts each separate element.

However,

> Compatibility Character is (1) A character encoded only for compatibility with preexisting character encoding standards to support transcoding. (2) A character that has a compatibility decomposition.

A precomposed character appears to be a compatibility character in sense (1), but not (necessarily?) in sense (2), because compatibility decomposition is different from 'ordinary' decomposition.