Accredited Standards Committee	**Doc. No.:**	**L2/01-210**
NCITS*	**Date:**	21 May, 2001
L2, Codes and Character Sets	**Projects:**	
	Reference:	
	Reply to:	Edwin Hart
To: L2, Unicode Technical Committee		Applied Physics Laboratory 11100 Johns Hopkins Rd. Laurel, MD 20723-6099 Voice: +1-443-778-6926 FAX: +1-443-778-1093 E-Mail: edwin.hart@ jhuapl.edu

Subject: Potential Requirements against SQL Standard

Action Requested: Consider and act as necessary

Recently someone said that a CD to the SQL standard was under ballot. Somehow the issue of binary sorting of text encoded in Unicode arose. I have not seen the SQL CD so I cannot comment on it. However, I do see potential user issues and wanted to validate them so that, if needed, L2 and the UTC could voice concerns to the SQL committee.

I believe that if user sees that a DBMS claims to use "Unicode" to store text data, then the user will expect that the binary ordering of the "Unicode" data would be the same from one DBMS to another. We have another related thread, so let me hit it before returning to the original statement. While I would expect that users would like to see text data correctly sorted according to a locale (Unicode Technical Report #10 and SC 22/WG22 sorting standard), I also would expect to see a lot of ordering/indexing based on the binary value of the encoded text. Because Unicode has different encoding forms (UTF-16, UTF-32, UTF-8) that have different binary values and ordering for Unicode characters, databases that use different Unicode encoding forms will have different binary orderings. While programmers need to be aware of such differences in encoding forms, most users will be unaware of this issue and will be surprised when two databases order the "Unicode" data differently using binary ordering. We need to avoid surprising users with unexpected behavior.

Ken Whistler (SyBase) and Jianping Yang (Oracle) have argued that database "binary" ordering works on the bits without concern for the underlying data representation (fixed, floating point (different flavors), 8-bit text, JIS/shift-JIS, Unicode-encoded text). Jianping further argues that to get consistent binary ordering of Unicode text across the various Unicode encoding forms, the DBMS needs to first transform the text into a sorting key. I think that the sorting key for binary sorting should be some "normal" encoding form, e.g., UTF-32. Moreover, this is no longer a sort of binary data in the strict SQL sense because obtaining the expected behavior first requires the DBMS to transform the raw text into sorting keys. Based on these comments, I believe that the requirements need to be restated as follows.

---

* Operating under the procedures of *The American National Standards Institute*
NCITS Secretariat, Information Technology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922 (Telephone: 202.737.8888 FAX: 202.638.4922)

1. We do not want to define yet another Unicode encoding form or variant of an existing encoding form.

2. Users would prefer to see text data ordered according to their individual cultural conventions.

    a. Users need to see a consistent default ordering of text encoded in Unicode regardless of the underlying Unicode encoding form (UTF-16, UTF-32, UTF-8).

    b. A default ordering for Unicode characters should likely be based on the Unicode scalar value (UTF-32) of each character.

I believe that the "Unicode Collation Algorithm", Unicode Technical Standard 10, covers all of these requirements. Until now, we have not discussed the implications on SQL. I believe that SQL can accommodate the requested behavior. However, SQL can use true binary ordering of the Unicode text only if the text uses the default coding form (item 2.b. above). Basically, users cannot depend on SQL ordering binary (raw) Unicode data (item 2.a. above). Instead users will need to depend on the DBMS first transforming the Unicode text into appropriate sorting keys, even for binary sorting.