Chapter 3

Two New Approaches to Multi-Codepage Capability

Unicode

Double-Byte Codepage

Unicode is a white-character codepage originally designed by members of the Xerox PARC. It is now integrated into the ISO standards (ISO-DIS 10646) by the recently founded *Unicode, Inc*. In their canonical form, characters consume four bytes. This form is called UCS-4 (Universal Coded Character Set-4). However, the most commonly used portion of the set is expected to be the *Basic Multilingual Plane (BMP)* and in it character entities take up the lower two bytes each. This form is called UCS-2. The initial version (2 bytes) of Unicode (ISO-DIS 10646) contains approximately 40.000 characters covering European, Asian ideographs, Middle Eastern, Indian, and other languages. For the definition of characters of the Arabic or Thai language where base letters can have a huge variety of diacriticals or other marks, the technique of combining characters is used. In this technique a code value for a character can have up to 8 bytes.

Coded Languages

The Unicode codepage defines codes for all European and some Asian languages and for many ideograms (symbols for terms) of the Chinese, Japanese and Korean languages (collectively: *CJK ideograms*). Also, special alphabets (e.g. OCR, math. + tech. symbols, ...) are contained.[5] If Unicode were used as the R/3 solution, then the R/3 system could remain a single-codepage system since all languages would be specified in one codepage.

Solution with Unicode?

Although Unicode seems to solve most problems on the first view, a closer look shows that an implementation with Unicode cannot be realized for several reasons. Some problems even cannot be solved with Unicode. This is a list of current arguments against Unicode:

- None of the today's databases supports Unicode. They support either single-byte or double-byte character sets.
- At the moment, no frontend software and no UNIX operating system support Unicode directly. Windows NT does support Unicode (it works internally with Unicode), but the adaptation of an application would require considerable effort.
- Currently, no printer supports Unicode.

[5] A Unicode overview can be found in the article "Welt der Zeichen"; c't 9/92, page 241 ff.

- The implementation of Unicode in R/3 would require significant effort and cannot be realized with the 3.0 release.
- Conversion of all texts and other data that are codepage-dependent would considerably raise the costs for:[6]
  - Storage (doubled size in database, every character needs two bytes),
  - Backup (doubled time and twice as much backup media),
  - Data transfer (doubled time),
  - Database reorganization (doubled time).
- Sorting with Unicode cannot be accomplished using the binary representation of the symbols, since the codes are not assigned according to a lexicographic sequence. Unicode therefore does not solve the problem of sorting key fields.
- There are discussions in progress on adding 30,000 ideograms to the Japanese alphabet. Therefore, the two bytes of Unicode may not be enough in the long term to represent all symbols.

[6] This argument generally holds for any double-byte storage mechanism for texts.

R/3 System with Multi-Codepage Capability

This section proposes a solution for an *R/3 system with multi-codepage capability*.[7] The solution requires changes in the current R/3 implementation as well as guidelines and conventions for the installation of such an R/3 system. Those guidelines must be followed by enterprises that want to install an R/3 system that has multi-codepage capability.

Common Character Set

In the solution for distributed systems in chapter 2, it was mentioned that global data uses a common character set. This prerequisite also has to be fulfilled by an R/3 system that has multi-codepage capability. The special replication mechanisms for multiple-system data transfer are not needed, however. Data fields that are used globally (i.e. that are represented everywhere) may only contain symbols which are mapped to the same value in all codepages (i.e. same position in codepage table). This *common character set* comprises in most cases the character set of the English alphabet (this corresponds to 7-bit US-ASCII).

Local Codepage

Data fields containing symbols of a local codepage can only be changed on an appropriate application server. In this context, ”appropriate” means an application server that uses the same local codepage or a codepage that is compatible to the local codepage. Here, ”compatible” means that there is a meaningful conversion from the one character set to the other. This is the case if most symbols from the first codepage can be translated to symbols in the second.[8] For example, EUC and S-JIS can be mapped into one another. It is possible to group codepages so that all members of one group can be converted from one to another. For every codepage employed in an R/3 system, an application server that uses this codepage must be installed.

Assignment to a Frontend

An R/3 user’s frontend server determines the codepage that is assigned to her or his R/3 frontend process (SAPGUI).[9] During logon, SAPGUI establishes a connection to an R/3 application server. The codepage conversion described above may be required for data transfer between these two components. No symbols may be misrepresented during this conversion and a unique 1-to-1 translation must be made.

Representation of Data

Symbols that cannot be represented by SAPGUI will be shown as special symbols (bars ▮, suns ✲, ...) by the windows system or are not shown at all. This is not critical, since no end user can expect to view all special symbols or symbols of a double-byte codepage everywhere Fig. 3-1, Fig. 3-2 and Fig. 3-3

---

[7] Multi-language system based on more than one codepage.

[8] Symbols that cannot be mapped are uniquely converted to unused positions in the target codepage.

[9] On a PC, there is only one codepage for the complete machine; on a workstation (with UNIX operation system), different processes can be assigned different codepages.

show an example of an R/3 screen displayed with different frontends. The screen of Fig. 3-1 is displayed by a German frontend, and German Umlauts are represented correctly but Japanese characters are not. Fig. 3-2 gives the corresponding representation of a Japanese frontend and Fig. 3-3 shows the screen on an English frontend. It is important that no symbols are destroyed, i.e. symbols that cannot be represented reasonably must be returned to the application server without changes (if they are not changed by the user deliberately). Other output devices (printer, file, ...) do not require a conversion of the local data.

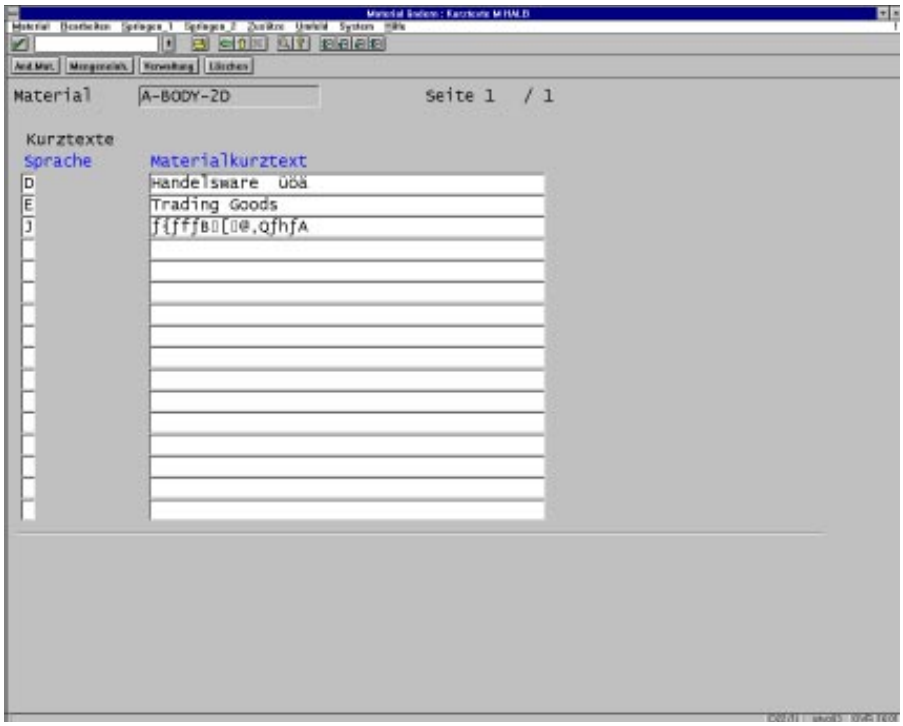

Fig. 3-1: Text Created with Several Codepages Displayed on a German Frontend

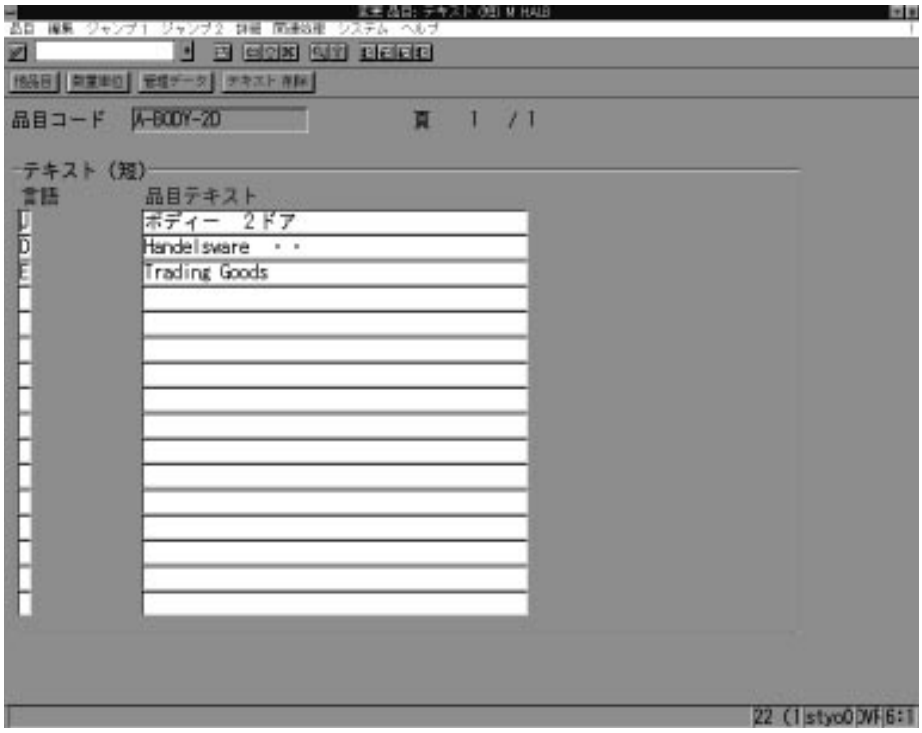

Fig. 3-2: Text Created with Several Codepages Displayed on a Japanese Frontend

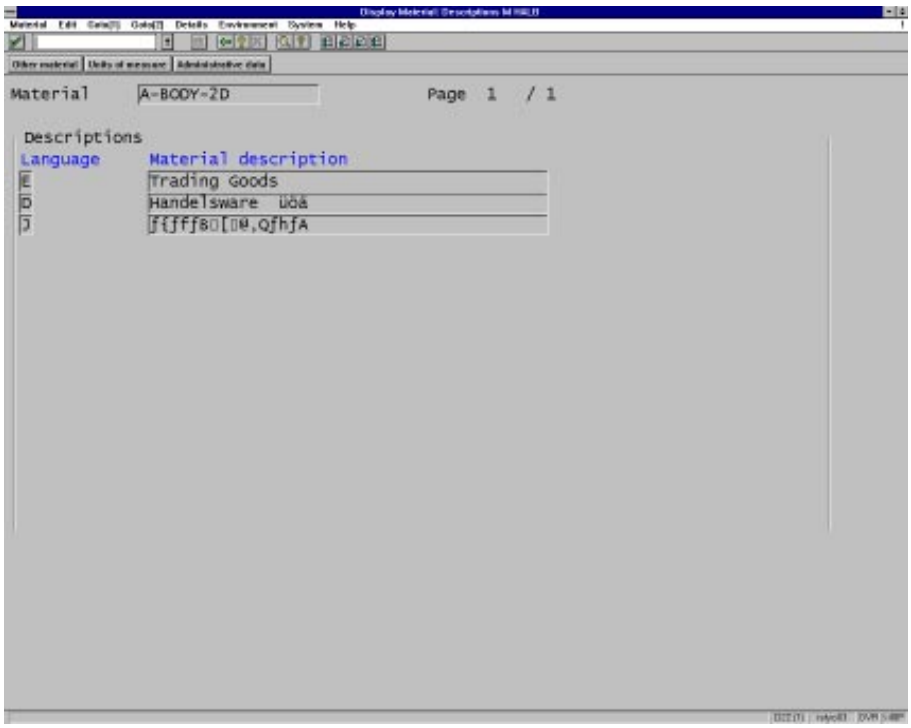

Fig. 3-3: Text Created with Several Codepages Displayed on an English Frontend

Printing

In printing, destruction of data cannot occur, since only output (i.e. read-only access) is done. There are printers that are capable of a dynamic codepage change. This functionality is assumed for printing from an R/3 system with multi-codepage capability. To be able to print texts containing symbols from several codepages, there will be a feature that allows the dynamic change of a codepage from an application program while printing (SAPscript command). In general, there is a considerable degree of organizational separation in print documents. Therefore, a mixing of codepages will rarely be necessary. During printing of ABAP lists in a spool work process, no information about the structure and codepage of the list is available. The codepage of the entire ABAP list is determined by the ABAP list processor in the dialog work process on the basis of the codepage of the application server.

Update Requests

For the update task service (independent process in the application server), there must be an update task server (i.e. an application server with update task service) active in the R/3 system for each codepage used. The dialog work processes of an application server will be assigned dynamically (starting with release 3.0) to the appropriate update task server. (In release 2.2x, the update task server will be determined at startup of the application server.)

Background Jobs

The update concept cannot be reused for background processing. That is, there cannot be a fixed relation between a background job and an application server with background work processes. Background jobs may include several single steps where each step can have its own country key. The codepage of the background process is changed dynamically if necessary for a step, and the background job can be processed in the correct codepage. A prerequisite for this procedure is the availability of all necessary codepages on an application server.

Communication with External Applications

Data transfer between systems is accomplished with communication mechanisms such as batch input, RFC or CPI-C. Data that must be imported from an external system usually contains only characters from the common character set. Because of this restriction, there cannot be a misinterpretation of data due to codepage mismatch. This guarantees an error-free communication between the systems. If the data contains national characters organizational measures must ensure that the data is imported into an application server that uses the appropriate codepage. These measures also apply to the export of data.