DB2 Codepage Umstellung

Was bei einer Umstellung auf Unicode zu beachten ist

Torsten Röber, SW Support Specialist DB2
April 2015

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Warum Unicode?

- Speicherung von XML Daten?
  - XML ‚native Storage‘ Tablespaces werden automatisch in UTF-8 angelegt
  - Auch in nicht mit Unicode (SBCS) angelegten DBs
- Speicherung von Daten unterschiedlicher Herkunft
  - Konvertierung in jede SBCS oder auch DBCS Codepage möglich
  - Kein ‘Verlust’ (Ersetzen) von Zeichen-Informationen innerhalb der DB
- Java-Zugriff ohne Konvertierung
  - Kommunikation immer in UTF-8

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Unicode Implementierung in DB2/LUW

- DB2 Linux, Unix & Windows verwendet Codesets UTF-8 und UTF-16
  - UTF-8 bedeutet 1 bis 4 Byte pro Zeichen
  - UTF-16 heißt genau 2 Byte pro Zeichen
  - Keine Speicherung von ‚Mixed Data', Ausnahme XML in SBCS-DBs
- Character-Datatypes (CHAR, VARCHAR, LONGVARCHAR, CLOB) werden in UTF-8 gespeichert
- Graphic-Datatypes (GRAPHIC, VARGRAPHIC) verwenden UTF-16
- Länge der Character-Datatypes wird in Byte spezifiziert
  - Verwendung von UTF-8 führt zu einer Erhöhung der Speicherlänge
  - Funktionen wie SUBSTR oder LENGTH sind per Default ‚Byte-orientiert'
- Daten werden im DRDA-Protokoll immer (!) vom Empfänger konvertiert
  - Daten einer Unicode-DB werden also in Codepage (oder CCSID) 1208 bzw. 1200 zum Client geschickt!

Unicode Implementierung in DB2/LUW

- Beispiel NAME (VARCHAR(128) in UTF-8) enthält 'Jürgen'
  - SELECT CHARACTER_LENGTH(NAME,CODEUNITS32)
    FROM T1 WHERE NAME = ’Jürgen’;
    oder
    SELECT CHARACTER_LENGTH(NAME,CODEUNITS16)
    FROM T1 WHERE NAME = ’Jürgen’;
    liefert den Wert 6
  - SELECT CHARACTER_LENGTH(NAME,OCTETS) FROM T1 WHERE NAME = ’Jürgen’;
    oder
    SELECT LENGTH(NAME) FROM T1 WHERE NAME = ’Jürgen’;
    liefert den Wert 7

Name	UTF-8 Representation	UTF-16 Representation	UTF-32 Representation
Jürgen	x'4AC3BC7267656E'	x'004A00FC007200670065006E'	x'0000004A000000FC00000072000000670000006500000006E'

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Umstellung einer Datenbank auf Unicode

- Der ‘Codeset’ wird beim Anlegen der Datenbank (CREATE DATABASE) spezifiziert
  - Ein Umstellung einer DB auf Unicode ist somit NICHT möglich!
- Eine neue UNICODE-Datenbank muss angelegt werden
  - Resourcen-Definitionen (Tabellen, Views, Indexes, etc.) können übernommen werden
  - Feldgrößen sind ggf. anzupassen (Umstellung auf VARCHAR?)
- Übernahme der Daten durch Export & Load/Import (db2move)
  - Export der Daten mit Unicode-Client erledigt die ‚Umstellung‘ beim Auslesen.
  - Import/Load arbeiten effizienter wenn die Daten nicht konvertiert werden müssen
  - Direkte Übernahme per Federation oder ‚Load from Cursor‘
- Replikation kann eine ‚Big Bang‘-Umstellung vermeiden helfen.
  - Bidirektionale Replikation ermöglicht die Koexistenz zweier inhaltlich weitgehend identischer Datenbanken.

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Wie umgeht man die Stolpersteine

- Vergrößerung der Feldlängen erfordert Umstellung der Anwendungen!
  - Begrenzung der Eingabefeldlänge in Hostanwendungen (EBCDIC = SBCS)
  - Hostvariablen entsprechend der Byte-Länge in der DB definieren.
  - Umstellung auf VARCHAR?
  - CODEUNITS16 oder CODEUNITS32 kann Datenbankweit verwendet werden
    - Reduzierung der maximalen Größe von Character-Feldern
    - Erhöhung des Speicherbedarfs auf Platte und im Bufferpool
    - Spaltenorientierte Tabellen (BLU) können nicht mit CODEUNITS16 oder CODEUNITS32 definiert werden.
- String-Funktionen erfordern ggf. Anpassungen
  - SUBSTR arbeitet mit Anzahl Bytes!
    - Konvertierung des ‚Strings‘ in CODEUNITS16 oder CODEUNITS32
  - Funtionen wie z.B. LENGTH erlauben die Angabe von CODEUNITS als Parameter

Wie umgeht man die Stolpersteine

- Manche Anwendungs-Server verwenden intern Unicode implementierung mit UTF-16, Daten werden aber immer im UTF-8 verschickt (‚Komprimierung').
  - JAVA-Anwendungen verwenden nur UTF-8
- Alle IBM Systeme haben eine Konvertierung zwischen Unicode und der lokalen SBCS Codepage implementiert.
  - Nicht alle Zeichen sind in einer SBCS-Codepage abzubilden!
  - Vorsicht! Bei ‚Round Trip'-Konvertierung droht Datenverlust

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Auswahl der ‚Collating Sequence' (Sortierfolge)

- Die ‚Collating Sequence' bestimmt die Sortierfolge von Zeichenfolgen
  - ORDER BY
  - Default ist eine binäre Sortierung
    - Unterschiede z.B. zwischen EBCDIC und ASCII bei Groß- und Kleinbuchstaben
    - Sonderzeichen und Umlaute sind nicht in ‚alphabethischer' oder ‚sprachlicher' Reihenfolge einsortiert!
- Es stehen verschiedene 'Collating Sequences' zur Verfügung
  - Sprachorientiert
  - EBCDIC-kompatibel
  - Benutzerdefiniert

Agenda

Warum Unicode?

Unicode Implementierung in DB2/LUW

Umstellung einer Datenbank auf Unicode

Wie umgeht man die Stolpersteine

Auswahl der ‚Collating Sequence‘

Haben Sie Fragen?

Haben Sie Fragen?

- UNICODE-Umstellung ist **nicht** trivial und sollte als umfassendes Projekt angegangen werden.
- Ist ein vollständige Umstellung unbedingt notwendig?
- Läßt sich eine schrittweise Umstellung mittels ‚Federation' oder ‚Replication' durchführen?
- Ein 100%-tiger Test der Anwendungen ist erforderlich!

- Haben Sie weitere Fragen?
  - torsten.roeber@de.ibm.com

Vielen Dank für Ihre Aufmerksamkeit!