Character Set Migration Best Practices

An Oracle White Paper
October 2002
Server Globalization Technology

Oracle Corporation

Introduction - Database Character Set Migration

Migrating from one database character set to another requires proper strategy and tools. This paper outlines the best practices for database character set migration that has been utilized on behalf of hundreds of customers successfully. Following these methods will help determine what strategies are best suited for your environment and will help minimize risk and downtime. This paper also highlights migration to Unicode. Many customers today are finding Unicode to be essential to supporting their global businesses.

Oracle provides consulting services for very large or complex environments to help minimize the downtime while maximizing the safe migration of business critical data.

Why migrate?

Database character set migration often occurs from a requirement to support new languages. As companies internationalize their operations and expand services to customers all around the world, they find the need to support data storage of more World languages than are available within their existing database character set. Historically, many legacy systems required support for only one or possibly a few languages; therefore, the original character set chosen had a limited repertoire of characters that could be supported. For example, in America a 7-bit character set called ASCII is satisfactory for supporting English data exclusively. While in Europe a variety of 8 bit European character sets can support specific subsets of European languages together with English. In Asia, multi byte character sets that could support a given Asian language and English were chosen. These were reasonable choices that fulfilled the initial requirements and provided the best combination of economy and performance. Today these choices are a barrier to supporting the global market necessitating character set migration.

Supporting new languages is not the sole motivation for character set migration. Cultural changes can also prompt support for a particular character or set of characters, such as the demand for the Euro Symbol throughout the world. Sometimes a changing operating environment is the deciding factor on the need to migrate character sets. Interoperability with environments which handle a different and non-compatible character set to the database character set can force the need for migration. Examples of this are databases that are exposed primarily to Unicode based Java clients or simply exposed to the World Wide Web where international users are providing character data for database storage. Even an English Windows client using code page 1252, supports many characters (‘€’, ‘™’) outside the repertoire of an ASCII database thus forcing the need for character set migration.

Another likely reason why migration may be required is consolidation of data stores in different character sets. An example of this could be two separate data stores, one holding French data and another holding Russian, that could be combined to a single data store in a character set which could represent both languages.

Choosing your new Character Set

Several character sets may meet your new language requirements. However, you should consider future language requirements as well when you choose the new database character set. If you expect to support additional languages in the future, then choose a character set that supports those languages to prevent the need to migrate again.

The database character set is independent of the operating system because Oracle has its own globalization architecture. The only restriction is that ASCII-based platforms do not support EBCDIC-based character sets and vice versa. For example, on an English Windows operating system, you can create and run a database with a Japanese character set. However, when the

client application accesses the database, it must be able to support the database character set with appropriate fonts and input methods. For example, you cannot directly insert or retrieve Japanese data on the English Windows operating system without first installing a Japanese font and input method. If you choose a database character set that is different from the character set of the client application, then the Oracle database must convert the application character set to the database character set.

For best performance, choose a character set that avoids character set conversion and uses the most efficient encoding for the languages desired. Single-byte character sets result in better performance than multibyte character sets, and they also are the most efficient in terms of space requirements. However, neither of these performance considerations may be practical as single-byte character sets limit how many languages you can support. Avoiding conversion may be impossible because some environments have multiple clients using different character sets accessing the same database.

Choosing the right character set requires consideration of all the client applications that will be accessing the database. The database character set should have equivalent characters in its repertoire otherwise data loss can occur. For example, if you are converting from a client with character set A to the database with character set B, then the destination character set B should be a superset of A. Oracle will convert any characters that are not available in character set B to a replacement character that is often specified as a question mark or as a linguistically related character. For example, 'ä' (a with an umlaut) may be converted to 'a'. If you have distributed environments, consider using character sets with similar character repertoires to avoid loss of data.

If all client applications use the same character set, then that character set or a superset of it is usually the best choice for the database character set. When client applications use different character sets, the database character set should be a superset of all the client character sets. This ensures that every character is represented when converting from a client character set to the database character set.

The application character set itself is usually the set of the characters that the application receives from its operating system's keyboard support and that it sends to its operating system's display support. On MS Windows, it may be either the active ANSI code page, the active OEM code page or Unicode depending on whether the application is an ANSI GUI, a command prompt or a Unicode application.

Some client applications perform internal character set conversion before sending characters to an Oracle database. An example would be a HTML browser that converts the O/S characters to the encoding of the processed HTML form before sending them to an application server. For the purpose of this discussion, the application character set in such case is the character set of data sent to Oracle libraries.

To summarize, choosing a character set requires combining the following elements to find the appropriate character set:

- Choose a character set that meets the current and future language requirements.
- The database character set should always be a superset or equivalent of all the collective native character sets of all the client applications that will access the database.
- The database character set on ASCII platforms must be a superset of US7ASCII and on EBCDIC platforms a superset of WE8EBCDIC37C.

Why Unicode might be the best choice

Supporting multilingual data often means using Unicode. Unicode is a universal encoded character set that allows you to store information in any language. Unicode provides a unique code point for every character, regardless of the platform, program, or language. With Unicode support, virtually all contemporary languages and scripts of the world can be easily encoded. This allows customers to host multiple languages in a single central database.

Unicode is fast becoming the de facto standard character set for emerging technologies and support is rapidly being adopted into new releases of software. It is no coincidence that a high proportion of character set migrations are from a legacy character set to Unicode as companies try to get ready for an anticipated influx of world data.
The character repertoire of Unicode virtually covers all characters in character sets supported by Oracle making it an ideal target for any character set migration, whether it is for consolidation or global readiness.
Generally, a Unicode database implemented in Oracle9*i* will provide about the same performance as a single byte database. Performance will be impacted greater on a Unicode database than a single byte database in environments where PL/SQL string manipulation functions are heavily used.

Unicode is a necessary requirement for certain language combinations. Typically, character sets support a subset of languages that originate from a specific writing script, such as Latin, Cyrillic, etc. When you introduce a mix of languages from different writing scripts then Unicode is usually required.

For more Information:

- Read the white paper: "**Oracle Unicode Database Support**" found on the Globalization Support Home Page:
    - http://otn.oracle.com/tech/globalization/content.html

Consider using the Unicode Datatype

An alternative to storing all data in the database as Unicode is to store only a subset, using the SQL NCHAR datatypes (NCHAR, NVARCHAR2, NCLOB). Unicode characters can be stored in columns of these datatypes regardless of the setting of the database character set. The NCHAR datatypes have been redefined in Oracle9*i* to be Unicode datatypes exclusively. In other words, they store data in the Unicode encoding only. The character set of NCHAR datatypes is called the National Character Set and it supports UTF-16 and UTF-8[1] in the following encodings:

- AL16UTF16 (default)
- UTF8

If you only need to support multilingual data in a limited number of columns, you can add columns of the SQL NCHAR datatypes to existing or new tables. You can also migrate specific columns from SQL CHAR datatypes (except CLOB) to SQL NCHAR datatypes easily using the ALTER TABLE MODIFY COLUMN command.

Example:

```
ALTER TABLE emp MODIFY (ename NVARCHAR2(10));
```

SQL NCHAR datatypes provide many benefits including:

[1] The Oracle character set UTF8 is Unicode 3.0 CESU-8 compliant. CESU-8 is identical to UTF-8 except for the way Unicode code points above U+00FFFF are encoded.

- You can rely on NCHAR datatypes to always be Unicode in order to build packaged applications that will be sold to customers.

- You can make your database environment match your native UCS-2 or UTF-16 application environment.

- You want the best possible performance - If your existing database character set is single-byte then extending it with SQL NCHAR datatypes may offer better performance than migrating the entire database to Unicode.

Migrating applications to support NCHAR is simplified because of the enhanced inter-operability of SQL NCHAR with other datatypes and character semantics. When interoperations between NCHAR and other datatypes are necessary, a user can either apply explicit or implicit conversions. By implicit and explicit conversions, a user can store, retrieve, and process NCHAR data the same way as CHAR data. Oracle9*i* supports implicit conversions between NCHAR and CHAR datatypes, as well as between NCHAR and other datatypes such as DATE, NUMBER, ROWID, RAW and CLOB. Implicit conversion is incurred whenever any inter-operation happens between different datatypes or the type of argument is different from the formal definition in a function. Explicit conversions are more controllable; users can decide which direction to convert. Oracle9*i* provides a wide range of conversion functions such as:

TO_NCHAR(), TO_CHAR(), TO_CLOB(), TO_NCLOB(), TO_NUMBER(), TO_DATE() UNISTR(), ASCIISTR(), ROWIDTONCHAR(), CHARTOROWID()

For more information:

- Read the white paper: "Migration to Unicode Datatypes for Multilingual Databases and Applications in Oracle9i" found on the Globalization Support Home Page:
    - http://otn.oracle.com/tech/globalization/content.html

Migration Concerns

Once it has been decided that a database should be migrated, an IT department typically has several other key concerns:

- Keep downtime window as short as possible during migration process. This means minimal time to perform the migration and a contingency plan should anything go wrong. The tolerated downtime can be anything from a matter of hours for mission critical applications, to several days for a common data store.
- Depending on the nature of the data, data loss can be a concern when performing migration. It is usually expected that the resulting data after migration be semantically equivalent to the data before migration, this can also extend to numeric data which references character data in some manner. For more information on Data Loss see the section Why do Invalid Data Exceptions Occur? The size of the resulting database should be estimated so that resources can be allocated for the migration to succeed. As migration to a multibyte character set will almost always result in an increase in size, data expansion is under consideration here.

- Performance of the database running with a new character set is also a consideration. Typically, character sets of similar properties perform equally well. So switching from one single byte character set to another, or one double byte to another will have no

performance impact. Switching from a single byte to a multibyte character set will have some performance impact.

- Introduction of a multibyte character set may break applications that were designed for single-byte character sets only. This may require a serious application overhaul.

Preparing the Database for Migration

Before proceeding with the migration it is essential to understand the scope of the conversion effort. Data analysis needs to occur to check for potential conversion problems and generally assess how much data actually needs to be converted.

The Character Set Scanner

No matter what migration method is used it is essential to first pre-scan the source data in order to assure successful migration. The Character Set Scanner is a powerful and sophisticated utility that scans all or specified character data fields and proactively identifies issues involved in migrating a database to a new database character set. The scan itself simulates the conversion of the existing data to a new character set and generates a summary report of the database scan. This report shows the scope of work required to convert the database to the new character set. It also helps determine the most appropriate database character set migration method. The report places each scanned data cell (e.g. a single column value in a particular row) into one of the following 4 categories:
Changeless data are data that do not change binary representation when converted from the source character set to the target. For example, if the letter 'A' contained in a cell has the binary code 0x41 in the source character set then it must have the same code 0x41 in the target character set to be considered changeless. A cell is changeless if all characters it contains are changeless. All data in the database must be changeless in order to safely use the ALTER DATABASE CHARACTER SET command.

Convertible data are that for which there exists a conversion path from the source character set to the target. For example, if the letter 'A' is contained in a convertible cell with a given binary code in the source character set (e.g. 0x41 in US7ASCII) then it must exist in the target character set as well, though its code may be different (e.g. 0xC1 in WE8EBCDIC37C). All data should fall under the convertible and changeless categories, with no potential truncation, before attempting to use the export and import utilities.

Truncation is where data resulting from conversion do not fit in the column's maximum length constraint. For example when migrating from a single byte Western European character set to Unicode, accented characters expand from 1 byte to 2 bytes, which may force the need for column expansion.

Data Loss (Lossy Conversion) is where there does not exist a conversion path to the target character set or the round-trip conversion gives data different to the original. For instance, if the Euro sign '€' is contained in a cell, encoded with binary code x80 in the source character set WE8MSWIN1252, then it cannot be converted to the target character set WE8ISO8859P1 asWE8ISO8859P1 does not have a binary code for this character.

Collectively, truncation and data loss cases are known as exceptions.

Data Truncation

The Character Set Scanner tests whether the post-converted data fits into the current column size. The Scanner pinpoints which columns need to be resized and displays the first 30 bytes within the column. Failure to modify the data or update the column sizes will cause data loss potentially for the entire table row during the Export and Import process. Data truncation must be considered in any scenarios where the migration target is a multibyte character set.

Typically, columns' maximum length constraints are expressed in bytes. A column defined as VARCHAR (1) will hold maximum one byte of binary codes. This is sufficient to store one character in any single byte encoding form but it may be insufficient to store one multibyte character. For example in Unicode UTF-8 a single character may take from one to –four bytes of storage.

The likelihood of truncation when migrating to UTF-8 often depends upon the language to be stored. For a database presumed to store only English data the chance of a cell truncating is very low where only a few symbolic characters will experience expansion from a single byte encoding to UTF-8. Western European languages contain more diacritics, which generally expand in UTF-8 to 2 bytes. Asian Languages are most likely to experience truncation because of the frequency of characters that expand when migrated from a legacy 2-byte character set to UTF-8 (typically 3 bytes).

Likelihood increases West to East

–English [£ ™ ©] (1%)
–European [ä ß ñ] (10%)
-Asian [丟 嫈 弯] (50%)

To avoid truncation problems users typically expand affected columns unless there is unexpected, undesirable data at the source causing the needed expansion at the target. In this case editing the undesirable data may be an option. If we resize the columns, we need to decide whether to expand to accommodate just the largest resulting cell or whether to expand according to the nature of the target character set. For example to migrate a CHAR(1) column in ISO 8859-1 to UTF-8 we could automatically expand it to a CHAR(4) to ensure it can always accommodate one character assuming that supplementary characters (having 4 bytes each) might be used later on. In cases where the SQL limit of the data type is exceeded, limitations must be placed on the storable data.

If the original copy of a database migrated through Export and Import utilities is not used after the migration, altering the column lengths can be done before the export step (but after taking a backup). This allows the Import utility to automatically create tables with correct column lengths and load data without truncation errors in a single run. Otherwise, tables have to be pre-created in the new database.

Changing Length Semantics

In single-byte character sets, the number of bytes and the number of characters in a string are typically the same. In multibyte character sets, a character or code unit consists of one or more bytes. Calculating column lengths in bytes is called **byte semantics**, while measuring column lengths in characters is called **character semantics**. Oracle9*i* allows column maximum length constraints to be expressed in character semantics, offering this way an alternative to redefining the byte sizes of columns. Length semantics are useful for both handling and defining the storage requirements for multibyte strings of varying width characters.

For example, you are migrating from an 8-bit Western European character set to a Unicode database (AL32UTF8). Suppose that you have existing VARCHAR2 columns that contain characters with diacritic marks. The scan indicates these columns would need to be resized to accommodate the expansion in the migration to a byte semantic multibyte database. Additionally you plan to add Asian data to your system. Using character semantics for your new migrated database may offer the best solution - particularly when the requirements for what type of information will be stored in these columns do not change. By changing a VARCHAR2(10 BYTE) column definition to a VARCHAR2(10 CHAR) definition you ensure that 10 characters will still fit in the column after migration even if they expand to more than 10 bytes of binary codes.

The NLS_LENGTH_SEMANTICS initialization parameter determines whether columns of character datatypes use byte or character semantics when they are created without an explicit qualifier. Byte semantics is the default for the database character set. Character length semantics is the default and the only allowable kind of length semantics for NCHAR datatypes. Length semantics can be explicitly specified in definitions of columns of CHAR datatypes by appending the BYTE or CHAR qualifier to the length value. The user cannot specify the qualifiers for NCHAR definitions.

Like with column length adjustments, if the original copy of a database migrated through Export and Import utilities is not used after the migration, altering the length semantics can be done before the export step (but after taking a backup). This allows the Import utility to automatically create tables with correct column lengths and load data without truncation errors in a single run. Otherwise, tables have to be pre-created in the new database.

Data Loss

The Oracle Export and Import utilities can convert the data from the original database character set to the new database character set. However, character set conversions can sometimes cause data loss or data corruption. Assurances must be made that the source data is clean and that each character has a comparable representation in the target character set. For example, if you are migrating from character set A to character set B, the destination character set B should be a superset of character set A. The destination character set B is a superset if it contains all the characters defined in character set A. Characters that are not available in character set B are converted to replacement characters, which are often specified as '?' or '¿' or a character that is related to the unavailable character. For example, 'ä' ('a' with an umlaut) can be replaced by 'a', '?' or '¿', depending on the replacement characters defined by the target character set.

Data loss can be broken up into the following three categories:**Data Loss at source** is where the source value (cell) contains binary codes that have no interpretation in the source character set, so no mapping to any target character set can be made. This can happen as a result of data corruption through incorrect string processing, or, more commonly, because some form of PASS-THRU scenario occurred and that data came in from some other character set but was not converted. Sometimes PASS-THRU is abused quite deliberately while other times it is not anticipated. For such data, character set and linguistic analysis should be applied to try to determine the anticipated character in the context of the data.

- **Data Loss at target** is where there exists no conversion path from the valid source character to the target encoding because that character does not exist in the target character set.
- **Data Loss in round-trip** occurs where there exists a mapping from the source character set to the target character set but converting the same code point back to the source character set yields a different character to the original.

For more information on Data Loss see the section "Why do Invalid Data Exceptions Occur?" below.

Why do Invalid Data Exceptions Occur?

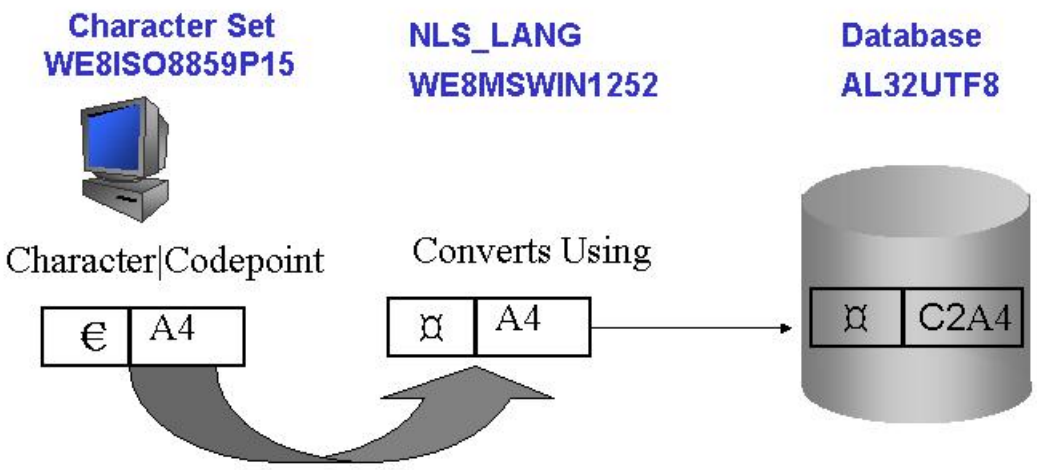
For data loss we need to analyze the reason why the loss occurred. Fixing the exceptions is referred to as data cleansing. Invalid data usually occur in a database because the NLS_LANG parameter is not set properly on the client and correct conversion does not take place. The NLS_LANG value should reflect the client application character set. In an English Windows environment, for example, the usual character set of GUI applications is the operating system code page 1252, which corresponds to the Oracle character set WE8MSWIN1252. When the NLS_LANG parameter is set properly, the database can automatically convert incoming data from the client operating system. When the NLS_LANG parameter is not set properly, then the data coming into the database is not converted properly. There are two forms of this type of bad conversion.

1. The first scenario is the NLS_LANG setting differs from the client application character set but also differs from the database character set.

    a. During the conversion process there may be characters in the client character set with binary codes having no interpretation in the NLS_LANG character set. Because there is no mapping for such characters to the database character set, replacement characters are used.

    b. During the conversion process there may be characters in the client character set with binary codes equivalent to some different characters in the NLS_LANG character set. So for example the Euro sign '€' could have the binary code 0xA4 in the client character set but this code could correspond to the general currency symbol '¤' in the NLS_LANG character set. In this case an incorrect conversion takes place and the data would be mapped to the encoding for the '¤' symbol in the database character set when the original character was meant to be an '€'. See Figure 1.

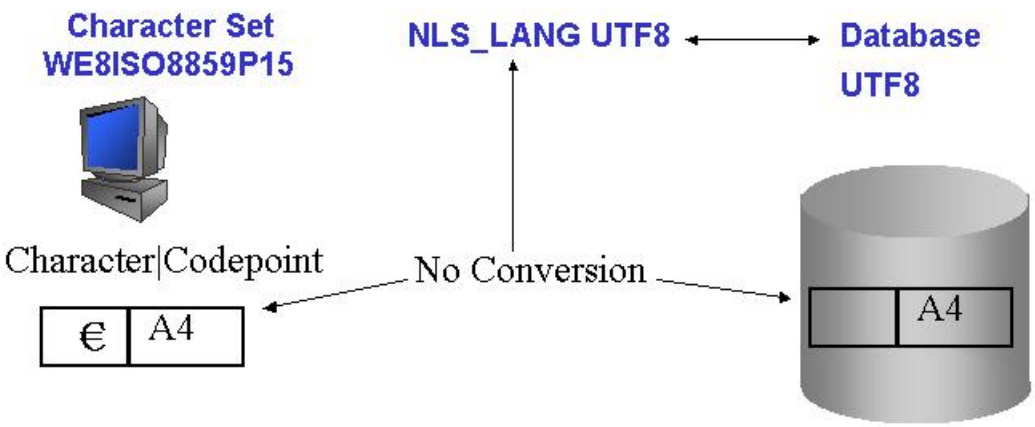
Figure 1



The Euro symbol is keyed in (codepoint 0xA4) but because the NLS_LANG is set to WE8MSWIN1252 codepoint 0xA4 is interpreted as a currency sign, which is then converted to the database character set where the currency sign is stored instead of the Euro.

2. The second scenario is where the NLS_LANG setting differs from the client application character set but matches the database character set. This often occurs because customers assume they should set the NLS_LANG on their client equal to the database character set, instead of the character set of the application. When the NLS_LANG setting matches the database character set, Oracle provides no conversion to maximize performance. This situation is referred to as PASS-THRU. The resulting unconverted data often have binary codes interpreted in the database character set as very different characters, effectively creating garbage data storage. A serious side effect of this scenario is that from the client application viewing perspective the data may appear to be fine because as the data is displayed no conversion takes place from the database back to the client, again because of the incorrect setting of the NLS_LANG. This PASS-THRU scenario masks improperly converted data stored in the database.

Figure 2



The Euro symbol is keyed in (codepoint A4) but because the NLS_LANG is set to UTF8 no conversion takes place and the character is stored as A4 which represents an illegal byte sequence .

The second scenario is common in many US7ASCII databases, where NLS_LANG is set to US7ASCII (or not set and thus defaulting to US7ASCII). Because many client applications of such databases send Windows code page 1252 or ISO 8859-1 (Latin 1) codes, the databases store illegal binary codes. The same scenario often happens with other client character sets, like Arabic (AR8MSWIN1256) or Japanese (JA16SJIS), sometimes quite deliberately.

- Most customers today find the need to migrate their legacy US7ASCII databases to a character set with a richer repertoire of characters that can support additional languages beyond English, or can support all incoming data from 8-bit client operating systems and applications. The actual migration step from US7ASCII to many ASCII based character sets could be simpler because the true subset/superset relationship allowed the use of the ALTER DATABASE CHARACTER SET command assuming the data were clean. However, in the scenario outlined above data do not correspond to the database character set declaration. The declaration must be corrected before the database can be migrated to, for example, Unicode. Oracle Migration Services should be considered because the task of identifying and fixing incorrect character set declarations is usually complex.

Conversion and Post-Conversion Tasks

Migration Methods

Oracle provides commands, utilities and services to help minimize the downtime while maximizing the safe migration of business critical data. Whatever migration method is used, it is first essential to pre-scan the data using the Character Set Migration Utility, as outlined previously.

Export/Import Utilities

The Oracle Export and Import utilities provide a simple way to transfer data objects (all or selected) between Oracle databases, even if they reside on platforms with different hardware and software configurations. When you run Export against an Oracle database, objects (such as tables) are extracted along with the data itself, followed by their related objects (such as indexes, comments, and grants),. The extracted data is written to an Export dump file. Export dump files can only be read by the Oracle Import utility. The Import utility reads the object definitions and table data from the Export dump file. It inserts the data objects into an Oracle database.

In Oracle9*i* the Export utility always exports user data, including Unicode data, in the character sets of the database. The Import utility automatically converts the data to the character sets of the target database. In Oracle8*i* the Export utility exports user data converting them from the database character set to the character set of the NLS_LANG of the Export session. The Import utility first converts the data to the character set of the NLS_LANG of the Import session and then converts them to the character set of the target database. Care must be taken that the character set of the NLS_LANG for Export and Import sessions contain all characters to be migrated. This character set is usually chosen to be either the source database or the target database character set and it is usually the same for both Export and Import sessions. This choice is recommended especially with multibyte character sets, which pose some restrictions on export files. The Oracle8*i* conversions to and from the NLS_LANG character set happen in Oracle9*i* for DDL statements contained in the Export file.

See the Oracle Database Utilities Guide for more details on Export and Import.

Using the ALTER DATABASE CHARACTER SET Statement

The ALTER DATABASE CHARACTER SET statement can be used only under special circumstances. The ALTER DATABASE CHARACTER SET statement does not perform any data conversion - it merely retags the database to the new character set. Therefore it can be used if and only if the new character set is a strict superset of the current character set data.

The new character set is a strict superset of the current character set if:

- Each and every character in the current character set is available in the new character set and …

- Each and every character in the current character set has the same code point value in the new character set. For example many character sets are a strict superset of US7ASCII.

Another restriction of the ALTER DATABASE CHARACTER SET statement in Oracle9*i* is that it can be used only when the character set migration is between two single-byte character sets or between two multibyte character sets. If the planned character set migration is from a single-byte character set to a multibyte character set, then either:

- Use the Export and Import utilities

- Contact Oracle Support. They can provide means to override ALTER DATABASE CHARACTER SET restrictions under special circumstances. Be prepared to provide a scan report using the Character Set Scanner demonstrating that all data from the current source database to be migrated to the planned target character set are changeless.

Combining the ALTER DATABASE CHARACTER SET with selective Export/Import

The ALTER DATABASE CHARACTER SET is the simplest way to migrate because it does not perform any data conversion. Often the majority of data in the source character set to be migrated is a strict subset of the target character set. However, any amount of non-changeless (convertible) data would prevent usage of the ALTER DATABASE CHARACTER SET. Since full Export and Import can be very resource and time consuming for large databases a hybrid solution can be used. Basically the small amount of convertible data is first exported into a dump file and removed from the source database leaving only changeless data, which can then be retagged to the target character set via the ALTER DATABASE CHARACTER SET command. The dump file containing the remaining data is then converted via the Import utility to the new character set of the target database. A case study illustrating the proper circumstances and technique for doing this can be found below (Case Study 3).

Using the SQL*Loader Utility

The ALTER DATABASE CHARACTER SET and Export/Import methods may be supplemented with use of the SQL*Loader utility. The Direct Path of the SQL*Loader is the fastest way to import huge amount of data into a database. SQL*Loader builds database blocks and writes them directly to database files, possibly in parallel sessions, bypassing the conventional SQL processing layer. SQL*Loader, like Import, supports character set conversion of data being loaded.

The drawbacks of using SQL*Loader are the need to create control files for all tables to be imported and the need to spool source data to text files. The source tables can be spooled by SQL*Plus or by custom OCI or Pro*C programs. The SQL*Loader utility is most convenient for migrating only the biggest tables that too big for the Import utility.

See the Oracle Database Utilities Guide for more details on the SQL*Loader utility.

Oracle Migration Services

Typically, character set migration requires data to be unloaded from a database of one character set into a database of another character set, with character set conversion taking place in-between.
The process is often time consuming, and for a complete migration the database has to be essentially built from scratch. During this time, no access to the database is allowed. It is also often required that the unloaded data be held in a second data store before being loaded into the new instance. For a 1-terabyte database, three terabytes or more may be recommended to perform the migration. Oracle Migration Services offers tools and expertise to minimize the time and resources necessary to complete a successful migration. Consultants can assess potential data exceptions, cleanse data and run special utilities to do inline migration.

To contact Oracle Migration Services send and email to: **infonls_migration_us@oracle.com**

Run a Backup

At this stage the source data has been scanned, readied and a migration strategy has been chosen, so now for the conversion itself. It is essential that a backup be taken of the production database right before the actual conversion. If data has been updated since the last scan then running the Character Set Scanner again is necessary.

Assure there is Enough Space Allocated

When you migrate from a single byte to a multibyte system there are likely to be greater storage requirements for your new database. Data expansion can be calculated based on the ratio of bytes between the source and target character sets. For example, migrating from a single byte character set to a fixed width multibyte character set would make the calculation straightforward based on the ratio for example 1 byte to 2 bytes. For a variable width multibyte character set the calculation may be more complex. For example with UTF-8, ASCII characters occupy 1 byte each. Accented Latin characters, Greek, Cyrillic, Arabic, and Hebrew occupy two bytes each. Other characters, including Chinese, Japanese, Korean, Indian, occupy 3 bytes each. And supplementary characters occupy 4 bytes each.

The space calculations should consider the percentage of text data among all database data. The percentage of CLOB data is also relevant because CLOBs are stored in the Unicode fixed-width UCS-2 encoding, if the database character set is multibyte. This may change estimates, especially for ASCII and Far Eastern characters.

Run a Rehearsal

A rehearsal on a test environment is highly recommended. The rehearsal will help get you familiar with all the steps, which will reduce production downtime. The rehearsal also allows you to gauge how long the production conversion will take and the resources needed in order to properly schedule downtime. This is also a good time to test both legacy and new applications to verify they work correctly with the new database character set.

Perform Conversion and Post-Conversion Validation Tests

Once the conversion is completed and the database is brought back up it is a good idea to run validation tests. Validation tests include verifying the data conversion is correct. If the Import utility is used, verify that there are no warning or exceptions in the log. Sample data can be visually inspected and the Character Set Scanner can be employed to check for exceptions. Test all production applications to make sure they are working correctly. Finally monitor the system for a period of time for problems and performance.

Case Studies

The case studies below are meant to illustrate key strategies, techniques, and decisions that must be made in managing a safe efficient character set migration.

1. Migrating a US7ASCII database to support more languages

A customer wishes to migrate their US7ASCII database to support new languages for Western European and Asian countries. The data feed into this database has come strictly from client English Windows Systems with the NLS_LANG set to AMERICAN_AMERICA.US7ASCII creating a PASS-THRU scenario. In spite of this, the customer is hopeful that most or all the data is purely ASCII. The customer runs the Character Set Scanner with source and target set as US7ASCII and AL32UTF8 respectively. Scanning detects exception data that the customer determines on further inspection to be letters with diacritics and special characters stored in the database invalidly. The customer must now determine the most appropriate action to take.

- If the 8 bit data is unimportant it could be cleansed by some combination of deleting or editing so, for example, replace character 'ä' with 'a'. Once the data is cleansed and verified to be totally *changeless* by rescanning the ALTER DATABASE CHARACTER SET command could potentially be used. Since the planned character set migration is from a single-byte character set to a multibyte character set then if this is an Oracle9*i* database, the customer should contact Oracle Support. They can provide means to override the single-byte to multibyte character set restriction under special circumstances.

- On the other hand if the 8 bit data needs to be retained, another approach could be taken. For the most part the data will need to be visually inspected to confirm its nature. Given that the 8 bit data came from English Windows in this scenario two steps are required to make this work. We must alter the database character set to WE8MSWIN1252. Then do a full export and import to properly convert all data to UTF-8. Truncation now becomes a possible issue because accented European characters take 2 bytes in UTF-8, so some resizing of columns may also need to be handled. If scanning shows all the data is convertible to the new character set, and no columns will be truncated it means that the data can be safely migrated using the Export and Import utilities.

2. Migrating from WE8ISO8859P1 to WE8ISO8859P15 to support the Euro Symbol

The scenario is to migrate a WE8ISO8859P1 (Latin 1) database to WE8ISO8859P15 (Latin 15) in order to support the Euro Symbol. WE8ISO8859P1 has a small subset of characters (‘¼’, ‘½’, ‘¾’, ‘¤’, etc.) not supported in WE8ISO8859P15. All other characters in WE8ISO8859P1 have a direct mapping to the same encoding in WE8ISO8859P15. If scanning shows all data to be changeless and the ALTER DATABASE CHARACTER SET command could potentially be used, the customer may contact Oracle Support for help in overcoming the standard superset restriction of the command. Otherwise a decision would have to be made on what to do with the non-convertible data.

3. Migrating from a Western European character set to AL32UTF8 to add support of Asian languages

The majority of databases using Western European character sets typically have a very high percentage of ASCII data. By selectively migrating only the data columns with Western European data and removing them from the database, we are left with the remaining data being ASCII, allowing us to potentially use the ALTER DATABASE CHARACTER SET. The benefit to this approach is that it can save a great deal of downtime in doing a full conversion using for example

the Export and Import utilities. Scanning will reveal the ASCII data as changeless. While the valid Western European data will be convertible.

- If the usage of Western European characters (accented letters and special characters such as the Euro which will convert to two bytes in UTF-8) turns out to be very high a full export or import may be the best option.

- If the usage of Western European characters turns out to be low and isolated say for example only in a couple of tables, a selective export and import of just those tables could be easily done with the Export and Import utilities, leaving the remaining database to be changed via the ALTER DATABASE CHARACTER SET command. Since the planned character set migration is from a single-byte character set to a multibyte character set then if this is an Oracle9i database, the customer should contact Oracle Support. They can provide means to override the single-byte to multibyte character set restriction under special circumstances.

- If the usage of Western European characters turns out to be not so isolated, for instance there are small amounts in many of the tables then Oracles Migration Services might be the best option. The inline migration utility mentioned earlier can utilize the output from a scan to convert only the Western European data leaving the remaining ASCII data to be retagged as AL32UTF8 via the ALTER DATABASE CHARACTER SET command.

With the previously mentioned choices in mind lets look at how invalid data affects these scenarios. A customer wishes to migrate their WE8ISO8859P1 (Latin 1) database to AL32UTF8 (UTF-8) in order to support Asian Languages in addition to the existing Western European Languages that are already supported. The customer runs the Character Set Scanner with source and target set as WE8ISO8859P1 and AL32UTF8 respectively. Scanning reveals that there are some Japanese characters stored in the database invalidly. The customer must now determine the most appropriate action to take. The customer realizes this data likely got into the database due to a PASS-THRU scenario. It is preferred to keep this data, as the direction is to now support Asian data, but how? Selective Export/Import is an option but there are technical issues. Since the database is labeled WE8ISO8859P1 then the Export will create a dump file with this character set and the Import will fail when converting to AL32UTF8. Oracle's Migration Services is an ideal solution to deal with these types of scenarios.

4. Customer needs to support Simplified and Traditional Chinese

Customer currently supports Simplified Chinese using the database character set ZHS16GBK. The customer now needs to support Traditional Chinese as well. Supporting more then one Asian language simultaneously will typically require Unicode. The customer is using an Oracle9*i* database and chooses AL32UTF8 in order to support the additional 94,140 supplementary characters that include many Asian ideographs. After scanning and doing any necessary clean up the customer uses the Export and Import utilities to convert the data. In this scenario the customer finds they must lengthen some of their columns to handle the expansion from 2 bytes per character in ZHS16GBK to 3 bytes in AL32UTF8.

Conclusion

Character set migration can be a challenging endeavor. Done hastily without understanding the issues and proper steps can lead to a perilous outcome. Taking proper precautions and using proven techniques and tools outlined in this paper will help ensure a safe and efficient migration. When in doubt at any step seek help via Oracle Support and Migration Services. Oracle has helped many customers successfully migrate their databases to new character sets. Successful migrations can help consolidate databases and support new languages and cultural symbols,

leading to more efficient operations and expansion of audience and services. Feedback to this paper can be left at our discussion forum on OTN at:

otn.oracle.com/forums/globalization.html