Character Set Migration Best Practices

An Oracle White Paper
January 2005

ORACLE

Character Set Migration Best Practices

INTRODUCTION

If your migration is to an Oracle9*i* or earlier database release it is recommended to read the previous version of this document entitled "Database Character Set Migration in Oracle9*i*". 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. Oracle also provides consulting services for very large or complex environments to help minimize the downtime while maximizing the safe migration of business critical data. This paper also highlights migration to Unicode. Many customers today are finding Unicode to be essential to supporting their global businesses.

WHAT'S NEW IN MIGRATION IN ORACLE DATABASE 10G

The CSALTER script is a DBA tool for special character set migration that comes as part of the Character Set Scanner Utilities. Similar to the now obsolete ALTER DATABASE CHARACTER SET command, CSALTER is to be used where the migration of the existing data is a proper subset of the new database character set. Details of using the CSALTER script are described in this paper.

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 initially 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 was 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 which are exposed primarily to Unicode based Java clients or simply exposure 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 as a means to consolidate 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. For example, on an English Windows operating system, you can create and run a database with a Japanese character set. However, when the client operating system accesses the database, the client operating system 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 an English Windows operating system without first installing a Japanese font and input method. Another way to insert and retrieve Japanese data is to use a Japanese operating system remotely to access the database server. If you choose a database character set that is different from the character set on the client operating system, then the Oracle database must convert the operating system 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 character set conversion may be impossible because most environments have multiple clients using different code pages accessing the same database.

Choosing the right character set requires consideration of all the client operating system environments and 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. It is also conceivable that if only a subset of characters from the repertoire of A are used that are a subset of B then character set A does not need to be a true subset of B for all the data to be convertible. This may be applicable where client and database character sets are very closely related such as WE8MSWIN1252 and WE8ISO8859P1. If characters like the Euro Symbol are avoided then conversion from WE8MSWIN1252 to WE8ISO8859P1 may be possible without lost data. 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.

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

1) Choose a character set that meets the current and future language requirements.
2) The database character set should always be a superset, or equivalent, of all the collective native character sets of all the client's operating system that will access the database.
3) 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.

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 develop, deploy, and 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.

Unicode is a character repertoire level superset of virtually all character sets supported by Oracle making it an ideal target for any character set migration, whether it is for consolidation or global readiness. For any new system deployment Oracle strongly recommends using a Unicode database.

Performance wise, a Unicode database implemented in Oracle Database 10g will run with up to 10% additional overhead than a single byte database. Performance will be impacted greater in environments where 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 Info

1) Read the white paper: "**Oracle Unicode Database Support**" found on the Globalization Support Home Page:
   - http://www.oracle.com/technology/tech/globalization/index.html

CONSIDER USING THE UNICODE DATATYPE

An alternative to storing all data in the database as Unicode, is to use the SQL NCHAR datatypes. Unicode characters can be stored in columns of these datatypes regardless of the setting of the database character set. The NCHAR datatype has been redefined in Oracle9*i* onward to be a Unicode datatype exclusively. In other words, it stores data in the Unicode encoding only. The National Character Set supports UTF-16 and [1]UTF-8 in the following encodings:

[1] The Oracle character set UTF8 is Unicode 3.0 UTF-8, CESU-8 compliant

2) AL16UTF16 (default)

3) 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 tables or new tables to support multiple languages incrementally. You can migrate specific columns from SQL CHAR datatypes 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 other benefits including:

1) You can rely on NCHAR datatypes to always be Unicode in order to build packaged applications that will be sold to customers.
2) You want your database environment to match your native UCS-2 or UTF-16 application environment.
3) 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 also 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. Oracle supports implicit conversions between NCHAR and CHAR data types, as well as between NCHAR and other data types such as DATE, NUMBER, ROWID, RAW and CLOB. Implicit conversion is incurred whenever any inter-operation happens between different data types 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. Oracle 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:

1) Read the white paper: "Migration to Unicode Datatypes for Multilingual Databases and Applications in Oracle9i" found on the Globalization Support Home Page:

- http://www.oracle.com/technology/tech/globalization/index.html

MIGRATION CONCERNS

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

2) Keep downtime window as short as possible during migration process. This means minimal time to perform the migration and execute 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.

3) 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 after image 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.

5) 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.

PRE-CONVERSION TASKS

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 Database 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 Database Character Set Scanner is a powerful and sophisticated utility that scans all character data fields, or selective data, 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 at the end it 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. From the results of a scan we can place each data cell into one of 4 categories.

Changeless data is data that does not change binary representation when converting from the source character set to the target. For example in the source character set the letter 'A' is a hex 41 and in the target character set the letter 'A' is also encoded in hex 41. Therefore 'A" is changeless from source to target. All data must be changeless in order to safely use the CSALTER script in Oracle Database 10g where normally allowed.

Convertible data is that for which there exists a conversion path from the source character set to the target. For example in the source character set the letter 'A' is a hex 41 and in the target character set the letter 'A' is encoded in hex 90. 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 a where conversion took place and the resulting data does not fit in the column definition. 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 is where there does not exist a conversion path to the target character set or the round-trip conversion is different to the original. For instance the € is represented as hex 80 in WE8MSWIN1252 but is not part of the repertoire of WE8ISO8859P1 (Latin 1). Collectively, truncation and data loss cases are known as exceptions.

Data Truncation

The Database Character Set Scanner tests whether the post-converted data fits into the current column size. The Scanner will pinpoint which columns need to be resized and display 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 row of data during the Export and Import process. Data truncation must be considered in any scenarios where the migration target is a multibyte character set. Typically column definitions are enumerated in BYTE semantics. A column defined as VARCHAR (1) will hold one character in any single byte encoding form but it is may not be sufficient to store one multibyte character. For example in Unicode UTF-8 one single character may take 1-4 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, which expand when migrating 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 [亩 卞 佯] (50%)

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'. Replacement characters are defined by the target character set. Data loss can be broken up into the following 3 categories.

1) **Data Loss at source** is where the value has no character representation in the source character set, so no mapping to the target character set exists. 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 and determine the anticipated character in the context of the data.

2) **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.

3) **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.

POST SCAN, PRE-CONVERSION TASKS

Once the Database Character Set Scanner has been run it is time to deal with any unexpected issues and decide on the right migration process.

Dealing with Exceptions

The first step is to consider those data cells that fall into one of the exception categories and make a decision on how to deal with them.

Truncation

For truncation, users typically expand the column 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 column 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 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 might be used later on. In cases where the physical limit of the data type is exceeded, limitations must be placed on the storable data.

Changing Length Semantics

Introduced in Oracle9*i*, length semantics offers an alternative to redefining the sizes of columns that need expansion for multibyte conversion typically to UTF-8. Length semantics are useful for both string handling and defining the storage requirements for multibyte strings of varying widths. 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**.

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 post-migrated database may offer the best solution. Particularly when a large portion of the columns would otherwise need to be expanded and the requirements for what type of information will be stored in these columns do not change.

The NLS_LENGTH_SEMANTICS initialization parameter determines whether columns of character datatypes use byte or character semantics when they are created. Byte semantics is the default for the database character set. Character length semantics is the default and the only allowable kind of length semantics

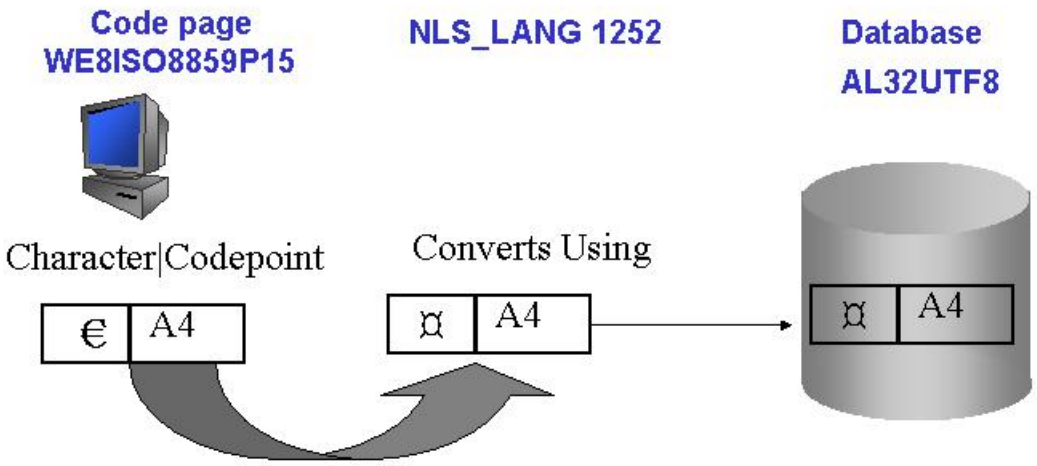
for NCHAR datatypes. The user cannot specify the CHAR or BYTE qualifier for NCHAR definitions.

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 occurs 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 operating system code page. In an English Windows environment, for example, the operating system code page is 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 OPERATING SYSTEM code page but also differs from the database character set.
    - During the conversion process there may be characters in the client code page with no equivalent value in the NLS_LANG character set. Because there is no mapping for such characters to the database character set, replacement characters are used.
    - During the conversion process there may be characters in the client code page with equivalent encoding but different values in the NLS_LANG character set. So for example the letter ‘A’ has the encoding hex 61 in the client code page but 061 is a ‘%’ symbol in the NLS_LANG character set. In this case an incorrect conversion takes place and the data is mapped to the encoding for the ‘%’ symbol in the database character set when the original character was meant to be an ‘A’. See Figure 1.

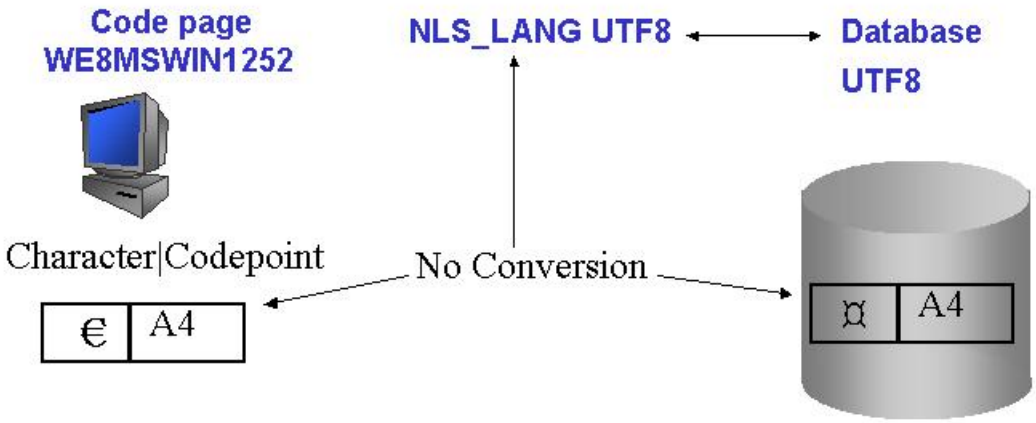
Figure 1



The Euro symbol is keyed in (codepoint A4) but because the NLS_LANG is set to 1252 codepoint A4 becomes 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 OPERATING SYSTEM code page but matches the database character set. This often occurs because customers assume to set the NLS_LANG on their client equal to the database character set, instead of the code page of the client OPERATING SYSTEM. When the NLS_LANG setting matches the database character set, Oracle provides no conversion to maximize performance referred to as PASS-THRU. The resulting unconverted data often has equivalent encodings in the database character set that map to totally different characters, effectively creating garbage data storage. A serious side affect of this scenario is that from the client operating system 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 the currency sign, instead of the Euro.

Special issues for US7ASCII databases

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 to support incoming data from client operating systems and applications that typically support 8 bit or multibyte data. The actual migration step from US7ASCII to many ASCII based character sets can be simpler because the true subset, superset relationship allows the use of the

ALTER DATABASE CHARACTER SET command in Oracle9*i* or the CSALTER script in Oracle Database 10g assuming the data is clean. But in many cases US7ASCII databases contain bad or invalid data. Again there are two common scenarios that cause this problem data:

1) The first issue outlined above, is called a PASS-THRU scenario. The NLS_LANG of client operating system that access the database have been incorrectly set to US7ASCII. Sometimes this is intentionally done to store 8 bit and multibyte data into the database and this case can be difficult to resolve without special expertise and care. Oracle Migration Services should be considered. Usually though the NLS_LANG has been unintentionally set incorrectly and some 8 bit data has crept into the otherwise 7 bit database. See Case Studies number 1 for an example of this scenario.

2) The second scenario is when the client NLS_LANG setting is set correctly. Typically Oracle will automatically convert from the client character set to the database character set as needed when the NLS_LANG indicates incoming characters may be of a different encoding. Since a US7ASCII database only supports 7 bit characters, only 7 bit characters are anticipated. Any 8 bit data that comes in, is converted into a replacement character, if possible a linguistically related 7 bit character otherwise a question mark is used. For example, ‘a’, a 7 bit character is linguistically related to the 8 bit character ‘ä’ (a with an umlaut).

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, followed by their related objects (such as indexes, comments, and grants), if any along with the data itself. 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 an Export dump file. It inserts the data objects into an Oracle database.

In *Oracle9i* onward, the Export utility always exports user data, including Unicode data, in the character set of the server. The character set is specified at database creation. In Oracle8*i* the Export utility exports user data, based on the NLS_LANG of the Export server. The Import utility automatically converts the data to the character set of the Import server. Care must be taken that the character set of the export file is a subset of the character set of the Import Server. There are restrictions on export files that are created from a multibyte character set. See the Oracle Database Utilities Guide for more details on Export and Import. Export/Import Utilities can be performed with little or no down time by using the features of Oracle Streams. See Oracle® Streams Concepts and Administration Guide for more details.

Using the CSALTER Script

The CSALTER script is a DBA tool for special character set migration that comes as part of the Character Set Scanner Utilities. The CSALTER script is the most straightforward way to migrate a character set, but it can only be used under special circumstances. The schema data of the database must be a strict subset of the new character set. The new character set is a strict superset of the existing data if:

1) Each and every character is available in the new character set.
2) Each and every character has the same code point value in the new character set. For example, many character sets are strict supersets of US7ASCII.

With the strict superset criteria in mind only the metadata is converted to the new character set by CSALTER with the following exception. The CSALTER script performs data conversion only on CLOB columns in the data dictionary and Sample Schema that have been created by Oracle. CLOB columns that users have created may need to be handled separately. Beginning with Oracle*9*i, some internal fields in the data dictionary and Sample Schema are stored in CLOB columns. Customers may also store data in CLOB fields as well. When the database character set is multibyte, CLOB data is stored in a format that is compatible with UCS-2 data. When the database character set is single-byte, CLOB data is stored using the database character set. Because the CSALTER

script converts data only in CLOB columns in the data dictionary and Sample Schema that were created by Oracle, any other CLOB columns that are created must be first exported and then dropped from the schema before the CSALTER script can be run.

Combining the CSALTER Script with selective Export/Import

The CSALTER script is potentially the fastest way to migrate because it does not perform any data conversion with the exception of CLOB columns in the data dictionary and Sample Schema that have been created by Oracle. Often the majority of data in the source character set to be migrated is a strict subset of the target character set. But any amount of non-changeless data (convertible) would prevent usage of the CSALTER script. 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 thereby allowing the CSALTER script to be run. The dump file containing the remaining data can then be 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. Please see the Oracle Database 10g Globalization Guide for more information on using the CSALTER script.

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 Database Character Set Scanner again is necessary.

Assure there is Enough Space Allocated

When you migrate from a single byte to a multibyte system there is 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 with multibyte character set the calculation would be 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 2 bytes each. Other characters, including Chinese, Japanese, Korean, Indian, occupy 3 bytes each. And supplementary characters occupy 4 bytes each.

Run a Rehearsal

A rehearsal on a test environment is also 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 warnings or exceptions in the log. Sample data can be visually inspected and the Database 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 Database 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 CSALTER script could potentially be used.

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 determine its nature. Given that the 8 bit data came from English Windows in this scenario, we could attempt use the scanner and choose a source character set that we think the data encompasses. One thing to note is that the Latin 1 character set WE8ISO8859P1and WE8MSWIN1252 are very similar but not the same. The Microsoft character set WE8MSWIN1252 has additional characters, which cover the entire range of 8 bit characters. Therefore setting the scanner to WE8MSWIN1252 will not help find 8 bit exceptions. It may be better in this case to scan setting the database source as WE8ISO8859P1and choosing the eventual new character set, for example AL32UTF8. 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. Two steps are required to make this work assuming the database character set is US7ASCII. We must run CSALTER script to change the database character set to WE8MSWIN1252 or WE8ISO8859P1. Then do a full export and import to properly convert all data to UTF-8.

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

The scenario is to migrate a WE8ISO8859P1 (Latin 1) database to WE8ISO8859P15 (Latin 9) 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 then the CSALTER script could be used. Note in the CSALTER script only checks that the existing schema data, not the character set is a strict subset of the target database character set. If scanning shows all data to not be changeless a decision would have to be made on what to do with the non-convertible data. Using the Oracle Locale Builder to customize the WE8ISO8859P15 character set adding the unsupported WE8ISO8859P1 characters then migrating via Export and Import to support these extra characters would be possible.

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 it from the database, we are left with the remaining data being ASCII, allowing us to potentially use the CSALTER script. 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 CSALTER script once scanning confirms the remaining data is changeless.

- 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 CSALTER script.

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 Database 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 Oracle Database 10g 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 serious 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:

http://www.oracle.com/forums/forum.jsp?forum=50

Character Set Migration Best Practices
January 2005
Author: Barry Trute

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com