

Volume 2009-08

Published: Aug. 21, 2009

www.DBA Online.org

Email: admin@DBA Online.org

Current Vol. Editor: Joanne Zou

DBA ONLINE

美國數據管理協會

Newsletter

*DBA Online – On the front line of database administration
DBA Online powers Oracle DBA*

DBA Online

Contents

Summary of DBA Online’s Aug 2009 Seminar	1
Migrate Database Character Set to Unicode.....	3

DBA Online’s August 2009 Seminar

This year is the 10th anniversary of American DBA Online. We have successfully organized an Oracle database seminar, as our 2nd event of the year, on Saturday, Aug. 15, 2009 at the Crown Plaza Hotel, Edison, New Jersey.

In this seminar, committee member Joanne Zou hosted the event.

First, Mr. David Wang, the president and founder of DBA Online organization, gave the opening remarks and highlighted DBA Online’s accomplishments. Then, he briefly went over the history of DBA Online and its current status. Finally, he introduced current IT industry trends and shared his vision on future development. He also thanked all the members for their continuing support during the years.

This seminar consisted mainly about knowledge sharing/transfer among members, anchored by three key sessions.

In the first session, Ms. **Yan Li**, a member of DBA Online, gave her presentation of “**Migrate Database Character Set to Unicode**”, where she shared her experience in migrating Database Character Set to Unicode. As multinational corporations continue to expand their operations around the globe today, they are faced with the challenges of centralizing database administration while maintaining the character sets of different countries in Database and support different Oracle database versions which are being used in different countries. The suggested solution to this challenge is to migrate the character set in Oracle software to Unicode. However, there are no guidelines to follow in the industry. Ms. Li successfully accomplished the migration through extensive research and experiments of her own. She shared her strategies and explained her

method of migrating character sets to Unicode. Because this is a relatively new topic for a lot of DBA's, her presentation generated a lot of interest among the audience, many of whom had questions in the Q&A session all of which Ms. Li satisfactorily answered.

Next, Mr. **Stan Ma**, an executive member of DBA Online Committee, discussed his topic, **“Database Performance and Tuning After 10g Migration and Upgrade”** where he shared his experiences in database performance and tuning. In 2007, Stan participated in the complete process of Oracle 10g migration and upgrade in a major financial firm -- from initial planning to software installation, software upgrade, until eventually ending with post-upgrade performance tuning. Performing all of these tasks one after another was definitely not easy and as if that was not enough, there were many also many challenges during the implementation of this project. To overcome the challenges and solve the problems, Stan did a lot of searches on the Internet; He came up with some queries which he used extensively to search for solutions. While doing so, he compiled several useful scripts that can be readily used as daily troubleshooting tools in Oracle Database administration. These queries, as well as Stan's knowledge in performance tuning, caught the attention of many in the audience.

Last, but not least, the last session took a special format – panelist discussion.

Five senior Oracle DBAs, all members of DBA Online, shared their knowledge/experience about real problems they solved on the job. This was a very interactive session with several audience members asking questions and in response panelists providing detailed answers/explanations. This format was very well liked by the audience because of these 5 expert panelists – Qi Wang, Jeff Zheng, Yan Lin, Herman Sun, and David Wang.

DBA Online would like to thank our sponsors, especially Oracle Corporation, for their continued support and confidence in us.

Thanks again to all speakers in the seminar! It was everyone's effort and involvement that made this DBA Online seminar another milestone!

DBA Online Committee

* * * * *

DBA ONLINE

MIGRATING DATABASE CHARACTER SETS TO UNICODE

=== * ===

Yan Li

INTRODUCTION

As companies internationalize their operations and expand services to customers all around the world, they often need to support data storage of more languages than is available within their existing database character set. Many companies today are finding Unicode to be essential to supporting their global businesses.

Based on a given character set and data, migration to Unicode can be rather simple or very complicated. It requires the proper strategy and tools. Not fully understanding the issues and the suitable converting steps can lead to an unrecoverable outcome.

What is Unicode? How do you choose the right one for the corresponding database character set? How do you install and use the Character Set Scanner? What are the migration methods? How do you resolve data dictionary and application issues? What is the meaning of “changeless”, “data loss” (lossy conversion), “truncation” and “convertible”? How should they be handled?

This paper uses WE8ISO8859P1 to AL32UTF8 migration as an example to answer the above questions and gives step by step implementation approaches.

ASSUMPTIONS:

Database Version:	10g (CSALTER is for 10g and up)
Source NLS character set:	WE8ISL8859P1
Target NLS Character Set:	AL32UTF8
CharacterSet Scope:	DATABASE level -
NLS_CHARACTERSET	

Note: The national character set NLS_NCHARACTER_SET is always in Unicode and can be used for NCHAR, NVARCHAR2, and NCLOB columns.

Scripts in this paper are for sample use only. Please test / verify /

modify them before using them in your environments.

STANDARD UNICODE AND ORACLE CHARACTER SETS

Unicode is a universal encoded character set that allows you to store information in ANY language. Unicode provides a unique code point for each character, regardless of the platform, program, or language. The most recent Unicode version 5.1.0 contains over 100,000 characters.

Before going into the character sets migration steps, let’s offer a brief introduction of Unicode and Oracle’s corresponding character sets.

Major Unicode encoding formats:

- UTF-8** 8-bit variable-width 1-4 bytes / character
 - used by UNIX platforms and HTML browsers
 - strict superset of ASCII

- UCS-2** 16-bit fixed-width 2 bytes / character
 - used by Java and Microsoft Windows NT
 - supports standard Unicode up to version 3.0 only

- UTF-16** 16-bit fixed –width 2 or 4 bytes / character
 - extension of UCS-2, support new standard Unicode versions
 - used by Microsoft 2000, XP, 2003, etc

There are also some less popular formats such as UTF-4, UTF-7, and UTF-32

Two types of the standard Unicode are used by Oracle databases:

UNICODE	ORACLE
UTF-8	UTF8 and AL32UTF8 (mainly for NLS_CHARACTERSET)
UTF-16	AL16UTF16 (mainly for NLS-NCHAR_CHARACTERSET)

Note: Standard Unicode has a dash (e.g. UTF-8) while Oracle character sets do not.

Oracle’s Unicode Character Sets:

UTF8 is an 8-bit variable-width encoding, each character has 1 to 3 bytes, supporting Unicode standards up to version 3.0 only.

AL32UTF8 is also an 8-bit variable-width encoding, with each character has 1 to 4 bytes, and will continue supporting future Unicode standards. AL32UTF8 is the Oracle database character set that is appropriate for XMLType data. Therefore, the best choice for a Unicode character set is AL32UTF8.

However, AL32UTF8 character set is NOT recognized by pre-9i clients/server systems. If you still have 8i (or older) servers and clients connecting to the 9i/10g system, then you must use UTF8 until you can upgrade the older versions to 9i or higher.

AL16UTF16 is a 16-bit fixed-width encoding, with each character having 2 or 4 bytes. It is not used as a normal database character set, but rather for a national character set. AL16UTF16 will also continue supporting future Unicode standards.

Oracle database and corresponding Unicode versions:

ORACLE CharacterSet	RDBMS Version	UNICODE	UNICODE-version	Bit	Byte
AL24UTFFSS	7.2 – 8.1	UTF-8	1.1		(obsolete)
UTFE	8.0 – 8.16	UTF-8	2.1	8	UTF8 for EBCDIC platform
	8.1.7 – 11g		3.0		
UTF8	8.0 – 8.1.6	UTF-8	2.1	8	variable-width 1-3 bytes
	8.1.7 – 11g		3.0		
AL32UTF8	9.0	UTF-8	3.0	8	variable-width 1-4 bytes
	9.2		3.1		
	10.1		3.2		
	10.2		4.0		
	11.1		5.0		
AL16UTF16	9.0	UTF-16	3.0	16	fixed-width, 2 or 4 bytes
	9.2		3.1		
	10.1		3.2		
	10.2		4.0		
	11.1		5.0		

There are two ways to use Unicode in an Oracle database: database level or type level.

1. **NLS_CHARACTERSET** is used for CHAR, VARCHAR2, CLOB columns, and determines database encoding.
2. **NLS_NCHAR_CHARACTERSET** is used for NCHAR, NVARCHAR2, NCLOB columns and is used on the column level. For instance, creating a table with column type as NCHAR, NVARCHAR2, NCLOB, etc.

This paper will concentrate on database level migration.

Most Western European and North American languages use US7ASCII or WE8 series character sets such as WE8ISO8859P1, WE8ISO8859P15, WE8MSWIN1252, etc. The migration steps are similar, but you must refer to the source character set related documentations to resolve specific problems. This paper uses WE8ISO8859P1 as an example to discuss the migration to Unicode AL32UTF8.

Note: In 11g, if you don't specify the character set for the CREATE DATABASE COMMAND, the default will be changed from US7ASCII to AL32UTF8.

PRE-MIGRATION PREPARATIONS

Before migration, check your inventory and clean up existing objects whenever possible.

1. Gather information about the current database environment:

You need to gather and analyze your current database environment including the current character set, the database storage size, installed Oracle products, and init parameters such as nls_length_semantics, job_queue_processes, aq_tm_processes, etc.

2. Check if there are any non-ASCII object names

You also need to check if any existing object, username, or password has used non-ASCII characters. The following query produces results for non-ASCII object names. You must rename them if any exist.

```
SELECT object_name
FROM   dba_objects
WHERE  object_name <>
       convert(object_name, 'US7ASCII');
```

You may use a similar method to check your usernames and passwords stored in the database.

3. Clean up unused or unwanted objects

If possible, it would be a good idea to take this opportunity to cleanup the schemas, users, tables and columns, as well as any other objects which are no longer used in your database. The migration tasks and the duration of the migration depend on the amount of data that needs to be converted in the database.

It is not required to use CSALTER to migrate a non-Unicode database to Unicode. If your database is small, you may create a new Unicode database and do a full export/import from the source database to the target Unicode database. However, if your database is large, a full export/import may be very time consuming, and you may consider using CSALTER with a partial export/import. You must also consider the percentage of data to be converted in your database to the total size of the database. If a large portion of data needs to be converted anyway, you may still consider a full export/import because the partial method may not reduce much of the downtime.

CSSCAN UTILITY

No matter what migration method is used, it is essential to pre-scan the source data beforehand in order to assure successful migration. The CSSCAN utility, also known as the Character Set Scanner, is a very useful utility for character set migration. During the migrating process, you may need to run it quite a few times at different stages.

Install CSSCAN

To install CSSCAN, run the script in
\$ORACLE_HOME/rdbms/admin/csminst.sql as sysdba.

<i>Note: You may want to modify the script to change the tablespace name, otherwise, SYSTEM tablespace will be used.</i>
--

The script will create a schema owner CSMIG and its objects such as tables (CSM\$...) and views (CSMV\$...). Each time you run CSSCAN, the tables will be replaced, therefore, only one set of scan result is kept in the database.

Running CSSCAN

Running CSSCAN is similar to running other Oracle utilities like

exp/imp. Use the following to get command line options:

```
csscan help=y
```

You can scan at the full database level, owner level or table level. You can also put your options into a parfile, just like using other Oracle utilities.

Before running the CSALTER script (to be discussed later) to convert, you must run a FULL scan. Keep this in mind: always run CSSCAN using “sys as sysdba”, not “system” or “csmig”, etc.

After installing CSSCAN, you may run a sample to get a taste of the utility and your data. You may want to run CSSCAN from your current character set to your target Unicode. (Consider this as an example to explain CSSCAN, don’t use this result for CSALTER at this stage yet).

```
csscan FULL=Y FROMCHAR= WE8ISO8859P1
tochar=AL32UTF8
```

Note: Be cautious if you use “CAPTURE=Y”, especially when “FULL=”. It will put a large amount of data into your database and onto your OS (the output files). This option captures every “convertible” row which is usually not necessary.

Understanding the CSSCAN output

Running CSSCAN will produce three output files with the extensions txt, err and out. If no file name is given, the default name is “scan”.

```
scan.txt file - summary of exceptions
scan.err file - detailed rows with ROWIDs
scan.out file - log of the scan process
                (e.g., ORA- error occurred during
                CSSCAN)
```

The following is a portion of the scan.txt file:

[Database Size]			
Tablespace	Used	Free	
Total	Expansion		

SYSTEM	12,586.25M	5,413.75M	18,0
00.00M	343.00K		
TSABCD	9,807.38M	2,192.63M	12,0
00.00M	13.38M		
TSXXX	267.44M	1,732.56M	2,0
00.00M	.00K		

TSYYY	64.00K	1,999.94M	2,0
00.00M	.00K		
TSZZZ	7,285.88M	714.13M	8,0
00.00M	.00K		
TSINDEX	640.06M	1,359.94M	2,0
00.00M	.00K		
TSUNDO1	2,141.13M	93,858.88M	96,0
00.00M	.00K		
TSUNDO2	24,519.00M	35,481.00M	60,0
00.00M	.00K		
... ..			

Total	3,581,509.00M	586,521.98M	4,168,0
30.98M	586.55M		

[Data Dictionary Conversion Summary]

Datatype	Changeless	Convertible
Truncation	Lossy	

VARCHAR2	136,913,559	0
0	154	
CHAR	7	0
0	0	
LONG	6,614,799	0
0	0	
CLOB	58	81
0	0	

Total	143,528,423	81
0	154	
Total in percentage	100%	0%
0%	0%	

[Application Data Conversion Summary]

Datatype	Changeless	Convertible
Truncation	Lossy	

VARCHAR2	189,317,244,480	0
14,326	7,455,330	
CHAR	977,410,022	0
0	0	
LONG	36,450,616	0
0	648	
CLOB	585,086,356	203,702
0	523	

Total	190,916,191,474	203,702
14,326	7,456,501	
Total in percentage	100%	0%

0%

0%

The database size section may help you to estimate the growth of the database after migration. The conversion summary section gives the total number and types of the data dictionary and the application data. The summary also tells you how many tables must be converted in your database. This information can help you to choose a migration method that minimizes the downtime.

The scan output can be queried from CSMIG owned tables (CSM\$) and views (CSMV\$). However, when you run CSSCAN again, the existing data will be replaced.

CSSCAN results in four kinds of data:

- 1. Changeless** - data that does not change binary representation when converted from the source character set to the target.
- 2. Convertible** - data for which there exists a conversion path from the source character set to the target.
- 3. Truncation** – data resulting from conversion that does not fit within the column’s maximum length constraint.
- 4. Data Loss (Lossy Conversion)** – data for which there does not exist a conversion path to the target character set or the round-trip conversion gives data different to the original.

Convertible data can be handled using a full or partial export/import (this is discussed later). Tuncation and data loss cases are known to be exceptions. You must address lossy and truncation issues regardless of using the full or partial method.

*Note: In order to use CSALTER, the CSSCAN output must be:
Changeless for all CHAR, VARCHAR2, and LONG data (Data Dictionary and user)
Changeless for all USER CLOB
Convertible and changeless for all Data Dictionary CLOB*

In other words, the following issues must be addressed before using CSALTER:

1. lossy conversion
2. truncation
3. data dictionary (lossy, truncation, convertible)
4. convertible

The “convertible” must be processed as the last step. All others can

be handled earlier. If the full export/import method is chosen, data dictionary issues from the source database can be ignored.

HANDLING LOSSY CONVERSION

What is “data loss” or “lossy conversion”?

Lossy data is not a valid code point for the source character set or the target character set. For instance, the Euro symbol € does not have a binary code for the WE8ISO8859P1 character set.

If no action is taken, then the “lossy” data will be “lost” in the conversion. Every "lossy" character will be converted to the same "default replacement character", usually as a question mark "?" or an inverted question mark "¿".

Note: If you have lossy data, never export/import to a database with a different character set. Once you have performed a conversion, there is no way to "recover" the original data. This can only be done with CSALTER.

A common cause of the “lossy” data is due to incorrect NLS_LANG client setting. In an environment using English or Western European languages, Windows clients’ NLS_LANG might have been set to WE8ISO8859P1. This is NOT correct for Windows systems. As a result, the actual WE8MSWIN1252 codes are stored in the WE8ISO8859P1 database in most cases.

Up to Oracle 8i, the default NLS_LANG of a Windows client installation was WE8ISO8895P1, which is in fact incorrect, as the correct value is WE8MSWIN1252. This should be corrected as soon as possible and must be addressed before converting to Unicode.

It is important to set NLS_LANG to its actual client’s operating system environment, not necessarily the database character set. The following registry value can be checked to determine Windows ACP setting:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Nls\CodePages\ACP
```

If the value is 1252, NLS_LANG should be set to WE8MSWIN1252.

If the source database has data that is NOT defined in the source character set, running CSSCAN with both the parameter FROMCHAR and TOCHAR defined to the current character set will result in ‘lossy’ data in the output.

```
CSSCAN FULL=Y FROMCHAR=WE8ISO8859P1
TOCHAR=WE8ISO8859P1
```

The following is a sample of ‘lossy’ data in the scan.err output:

```
ROWID                Exception Type          Size Cell Data(f
irst 30 bytes)
-----
AACQtnAC3AAAp2XAAF  lossy conversion          XT-Q QVRLTQ
AACQtnAC3AAAgHsAAG  lossy conversion          VO-C KTLUOC
```

what can we do?

Can “lossy” be fixed? How? Under what conditions?

Some lossy data can be resolved by converting the character set to its “strict” superset. For instance, all characters included in the WE8ISO8859P1 character set are defined in WE8MSWIN1252 with the same code point, which means that WE8MSWIN1252 is a binary or "strict" superset of WE8ISO8859P1 (as well as US7ASCII).

Note: You can create a database on UNIX with a "Windows" character set like WE8MSWIN1252. Oracle does not depend on the OS character set for the DATABASE (or national) character set.

Oracle recommends using Unicode whenever possible. Nevertheless, the recommendations for some popular database character sets are:

```
US7ASCII:           better to migrate to WE8MSWIN1252, or
WE8ISO8859P15 etc.
WE8ISO8859P1:       better to migrate to WE8MSWIN1252
UTF8:               better to migrate to AL32UTF8
```

To determine whether or not converting to WE8MSWIN1252 can resolve the lossy data of the current WE8ISO8859P1 database, run the following CSSCAN:

```
csscan FULL=Y FROMCHAR=WE8MSWIN1252
TOCHAR=WE8MSWIN1252
```

If you generate the following 3 lines in the summary output file (.txt), then running CSALTER can resolve the “lossy” issue:

```
All character type data in the data dictionary remain the same in
the new character set
All character type application data remain the same in the new
character set
The data dictionary can be safely migrated using the CSALTER
script
```

Converting WE8ISO8859P1 to WE8MSWIN1252 using

CSALTER

Now it is time to get rid of the ‘lossy’ data. Make sure your last CSSCAN’s FROMCHAR and TOCHAR are both WE8MSWIN1252. CSALTER is based on your last CSSCAN’s results.

Note: Although we are converting from WE8ISO8859P1 to WE8MSWIN1252, the CSSCAN’s FROMCHAR and TOCHAR are both WE8MSWIN1252. Otherwise, the 3 clean lines won’t be generated in order to use CSALTER.

CSALTER script is a PL/SQL block called csalter.plb located in \$ORACLE_HOME/rdbms/admin. This script comes with 10g. Before 10g, to change the database character set, you must use “ALTER DATABASE CHARACTER SET ...” and must handle some of the steps manually.

Before running CSALTER, be sure to backup the database, then shutdown listener and any applications connected to the database, and purge dba_recyclebin. Then run CSALTER as sysdba.

```

SELECT * from nls_database_parameters
WHERE parameter like '%CHARACTERSET%';

shutdown immediate
startup restrict

SPOOL To_WE8MSWIN1252.log
@$ORACLE_HOME/rdbms/admin/csalter.plb
SPOOL OFF

-- change back job_queue_processes/aq_tm_processes to
  original values

shutdown
startup

SELECT * from nls_database_parameters
WHERE parameter like '%CHARACTERSET%';

```

Your database character set is now WE8MSWIN1252 and the lossy data should be gone.

Note: The inverse operation WE8MSWIN1252 to WE8ISO8859P1 (or WE8MSWIN1252 to US7ASCII) is normally NOT possible without losing data.

HANDLING TRUNCATION DATA

Once we’ve gotten rid of “lossy” data, we can continue to the next

step of Unicode migration.

First, do a CSSCAN from your current character set (for our example, WE8MSWIN1252) to the Unicode chosen (e.g. AL32UTF8).

```
csscan FULL=Y FROMCHAR=WE8MSWIN1252  
TOCHAR=AL32UTF8
```

Most likely you will get “truncation” and “convertible” data now, especially if your database has non-ASCII characters.

What is truncation?

Typically, a column’s maximum length constraint is expressed in bytes. A column defined as VARCHAR(1) will hold a one byte of binary codes. This is sufficient for storing one character in any single byte encoding form but it may be insufficient for storing a multi-byte character. In Unicode, a single character may take anywhere from 1-4 bytes of storage.

In our WE8MSWIN1252 database now, every character is 1 byte, therefore, 1 char = 1 byte. However, because Unicode is variable-width, the same character may become 2, 3 or 4 bytes. Again, using the Euro symbol € as an example, in WE8MSWIN1252, its code is 1 byte while in AL32UTF8, it will use 3 bytes to store one Euro symbol. Supposing you use a column with VARCHAR2(1) to store it, it is like trying to put a 3-inch block into a 1-inch box. In order to fit into the box, the block will be truncated.

If you don’t adjust the data, you will get ORA-12899 error: “value too large for column” when you import the data.

How do you resolve truncation issues?

There are a few methods:

1. Enlarge the column using more bytes.

In our example, modifying VARCHAR2(1) to VARCHAR2(3) can resolve the problem. However, this method is not logical as we only need 1 character.

2. Change length semantics at the database level

Changing database parameter NLS_LENGTH_SEMANTICS to “CHAR” is another solution:

```
ALTER SYSTEM SET nls_length_semantics=CHAR  
scope=both;
```

However, some Oracle installed products such as ORACLE TEXT (context index, etc.) won't support instance level "CHAR" length semantics.

The instance or session value will only be used when creating NEW columns. Setting NLS_LENGTH_SEMANTICS to CHAR will NOT adapt current existing column definitions. In other words, if you have columns using BYTE now and you change the instance parameter to CHAR, then those columns will still be BYTE. To change existing tables you must use "alter table".

Note: Don't create a database with NLS_LENGTH_SEMANTICS=CHAR, you will have to change it after database is created because the data dictionary must use BYTE length semantics.

3. Change length semantics at the column level

You may use "alter table" SQL scripts to modify the CHAR and VARCHAR2 columns (all columns or just the ones with 'truncation' data) from BYTE to CHAR. This can be done before the export step and allows the import utility to automatically create tables with correct column lengths and load data without truncation errors in a single run.

Although it is enough for the actual conversion to take action only on the columns reported, it is strongly recommended to use CHAR semantics for all columns when transferring to a variable-width character set like AL32UTF8.

The following can be used to generate script to change all CHAR and VARCHAR2 columns from BYTE to CHAR semantics:

```
SET pagesize 0 linesize 120 feedback off verify
off
SPOOL byte_to_char.sql

SELECT
'alter table '||t.owner||'.'||t.table_name||
' modify ('||c.column_name||
' ||c.data_type||'('||c.data_length||
' CHAR));'
FROM dba_tab_columns c, dba_tables t
WHERE c.owner=t.owner
AND t.owner not in ('SYS','SYSTEM','SYSMAN',...)
      -- exclude system installed owners
AND c.table_name=c.table_name
AND c.char_used = 'B'      -- only if not in CHAR
```

```

semantics
AND t.PARTITIONED !='YES' -- exclude partitioned
tables (doc 330964.1)
AND c.table_name not in (select table_name from
dba_external_tables)
-- exclude external tables
AND c.data_type in ('VARCHAR2', 'CHAR');

SPOOL OFF

```

Note: Byte semantics is the default for the database character set. Character length semantics is the default and the only allowable length semantics for NCHAR data types.

4. Using export/import

This is also a column level change. First, export the tables and then drop them. Second, alter the session set `NLS_LENGTH_SEMANTICS=CHAR;` and re-create the tables with no data. Lastly, import data with `IGNORE=Y`.

If you have partitioned tables with CHAR semantics, you must fix them manually (see Doc 330964.1). Also, if you have functional indexes on affected columns, you must drop them before changing to CHAR semantics and then recreate them afterwards.

How do you handle data exceeding maximum bytes?

What if a VARCHAR2 column stretches beyond 4000 bytes after changing to CHAR length semantics? Modifying the column to VARCHAR2(4000 CHAR) will not work since Oracle's maximum length for VARCHAR2 is 4000 bytes (not chars)! This problem also exists for CHAR type exceeding 2000 bytes.

Single byte character set:	1 byte / char
UTF8	1-3 bytes / char
AL32UTF8	1-4 bytes / char

As a result, 2000 bytes can hold a minimum of 500 (2000/4) to a maximum of 2000 characters, while 4000 bytes can hold a minimum of 1000 (4000/4) to a maximum of 4000 characters for AL32UTF8.

Note: It's rare to have a 4-byte character. Examples are some old Chinese characters.

Suppose you have a table named RECORDS which has a column named DESCRIPTION as VARCHAR2(4000) and you see the following in the CSSCAN output:

```
User : ABC
```

```

Table : RECORDS
Column: DESCRIPTION
Type  : VARCHAR2(4000)
Number of Exceptions  1
Max Post Conversion Data Size: 4082

```

And you find that the ROWID is
'AACQtnAC3AAAp2XAAF'.

You can't put 4082 blocks into a box with a maximum of 4000 cells. Now what should you do?

1. Shorten the violated data:

```

UPDATE ABC.RECORDS
SET  DESCRIPTION = 'abcdefg...' -- a value <= 4000
bytes
WHERE ROWID = 'AACQtnAC3AAAp2XAAF';

```

But if you are not allowed to change the data or you have too many violated rows in the column, you should consider using CLOB instead of VARCHAR2.

2. Change the VARCHAR2 column to CLOB

You can not simply alter a VARCHAR2 column to CLOB. You may need to do the following:

```

ALTER TABLE ABC.RECORDS ADD (tmp CLOB);
UPDATE ABC.RECORDS SET tmp=TO_CLOB(DESCRIPTION);
COMMIT;
ALTER TABLE ABC.RECORDS DROP COLUMN DESCRIPTION;
ALTER TABLE ABC.RECORDS RENAME COLUMN tmp to
DESCRIPTION;

```

Note: If you change the column type to CLOB, don't forget to change the dependent objects, such as related PL/SQL packages, application programs, etc.

Other objects related to length semantic change

If a column is modified from BYTE to CHAR, you must consider other dependencies such as user defined types, stored procedures, functions, packages and other PL/SQL objects.

Using a stored procedure as an example, there may be code defined as a column VARCHAR2(5) which means VARCHAR2(5 BYTE). Now that the table column definition is changed to VARCHAR2(5 CHAR), you may have to make the PL/SQL object change corresponding to the table change.

There are at least two methods to do so:

1. Use explicit CHAR semantics in code:

Define variables explicitly with ‘CHAR’

```
x CHAR(5 CHAR);      -- not implicitly as x
CHAR(5);
y VARCHAR2(10 CHAR); -- not implicitly as y
VARCHAR2(10);
```

This method is preferred to avoid confusion.

2. Change the session length semantics setting:

If for some reason, you cannot change the code explicitly, you may alter the session to use “CHAR” length semantics before compiling the PL/SQL objects. For example:

```
alter session set NLS_LENGTH_SEMANTICS='CHAR';
Create or replace procedure XYZ...
```

The following query can be used to find out what length semantics a PL/SQL object (procedure, function, package/body, type/body, trigger, etc) was compiled with:

```
SELECT owner, type, name, nls_length_semantics
From DBA_PLSQL_OBJECT_SETTINGS
Where owner = 'ABC'
And name = 'XYZ';
```

HANDLING DATA DICTIONARY ISSUES

Let’s discuss data dictionary issues before conversion. If you choose to perform a partial export/import with CSALTER, you must fix data dictionary issues before running CSALTER.

If you choose a full export/import into a pre-created Unicode database, you can ignore data dictionary issues from your source database as your data dictionary will be from the pre-installed Unicode database.

As mentioned before, in order to use CSALTER, all data resulting from the FULL CSSCAN must be “changeless” or “convertible” CLOBs in the data dictionary. The “convertible” CLOBs in the data dictionary can be handled by using CSALTER in 10g and up.

The following query gives a list of data dictionary objects you must address manually before using CSALTER:

```

SELECT  table_name, column_type, coumn_name,
        conv_rows, exceed_size_rows,
        data_loss_rows
FROM    csmig.csmv$columns
WHERE   owner_name='SYS'
AND     (conv_rows>0 or exceed_size_rows>0 or
        data_loss_rows>0)
AND NOT (column_type = 'CLOB' and conv_rows>0);

```

You may face different kinds of data dictionary issues depending on your system data and the installed products. You must address them on a case by case basis. There exist many Oracle documents and workarounds regarding data dictionary issues. If you cannot find answers or if you are not sure about the resolutions, please open a Service Request with Oracle Support.

Here are some of the possible data dictionary issues and workarounds:

SYS.JOB\$

Remove the problematic jobs and resubmit them after conversion.

AWR related issues (SYS.WRH\$ and/or SYS.WRI\$)

Stop 'GATHER_STATS_JOB' from the job scheduler and drop the AWR snapshots.

```

SET pagesize 0 linesize 130 feedback off
verify off
SPOOL drop_snapshot.sql
SELECT
'execute
dbms_workload_repository.drop_snapshot_range' ||
min(snap_id) || ', ' || max(snap_id) || ');'
FROM SYS.WRH$_SQLTEXT;
SPOOL OFF

```

Then run the generate script to drop the snapshots.

Remember to get fresh statistics after the conversion.

SYS.HISTGRM\$

Delete statistics on the tables reported by running the following:

```

$ORACLE_HOME/nls/csscan/sql/analyze_histgrm
.sql

```

You may use the following to generate the script to delete the

statistics.

```

SET pagesize 0 linesize 120 feedback off
verify off
SPOOL delete_histgrm_stats.sql

SELECT DISTINCT
'ANALYZE TABLE
'||owner||'. '||object_name||' delete
statistics;'
FROM dba_objects, SYS.HISTGRM$
WHERE object_id = obj#
AND owner not like '%SYS%'
AND owner not in ('XDB','MDSYS','OUTLN')-
- exclude system owners
AND SYS.HISTGRM$.ROWID IN
(SELECT data_rowid
FROM csmv$errors
WHERE
owner_name||'. '||table_name='SYS.HISTGRM$'
AND error_type in
('CONVERTIBLE','DATA_LOSS'));

SPOOL OFF

```

Then run the generated script to delete the troubled statistics. You may re-analyze them after converting to Unicode.

SYS.SOURCE\$

Analyze and investigate the output from running the following:

```

$ORACLE_HOME/nls/csscan/sql/analyze_source.
sql

```

You may want to drop the objects, then fix and recompile them later in the Unicode database.

Data dictionary problems can be very sensitive. The above cases may or may not work for your specific situations. Be cautious when making changes. When in doubt, open a SR with Oracle.

HANDLING CONVERTIBLE DATA

Now our journey is approaching to the destination – Unicode.

The “convertible” data is valid, but the characters will change to a different code point in the new character set. For example, the pound sign £ is "code 163" (A3 in hex) in the WE8ISO8859P1 and WE8MSWIN1252 character sets, but in AL32UTF8 it is code 49827 (C2 A3 in hex).

When using CSALTER, any application data that is "convertible" needs to be exported before changing the character set and imported afterwards.

Oracle suggests not using data pump (expdp/impdp) when converting to Unicode or other character sets on all 10g versions lower than 10.2.0.4 (and 11.1.0.6). It will provoke data corruption unless you applied Patch 5874989 on the impdp side (expdp is not affected). The "old" exp/imp tools are not affected.

You may choose one of these two methods to handle the convertibles:

Method 1: Full export / import (exp / imp)

1. Pre-create a Unicode database (AL32UTF8)
2. Set NLS_LANG to your current database character set and perform a full export

```
NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252
exp FULL=Y ...
```

3. Set NLS_LANG to the character set you're exporting from and import to your Unicode database

```
NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252
imp FULL=Y ...
```

This is the traditional method. However, if your database is large, this method can be very time consuming. This paper will focus more on the next method.

Method 2: Partial export/import with CSALTER

- 1. Export the convertible tables with NLS_LANG set to current character set**

You may want to generate scripts for truncating tables, export / import parfiles, disable/enable triggers, foreign keys etc for later use. You can use CSMIG owned objects to do so based on your last CSSCAN results.

<i>Note: when you re-scan or get table structure changes after last CSSCAN, your CSMIG tables/views can be changed.</i>

To generate export parfile for the convertible tables of a schema:

```

SET echo off feed off lines 100 pages 0
SPOOL exp_conv_tab.&1..par
SELECT
  'files=exp_conv_tab.&1.dmp' || chr(10) ||
  'log=exp_conv_tab.&1.log' || chr(10) ||
  'buffer=100000' || chr(10) ||
  '...' - (other options if needed)
FROM dual;

SELECT 'TABLES='owner||'.' || table_name
FROM   dba_tables
WHERE  owner || table_name IN
      (SELECT distinct owner_name || table_name
       FROM csmig.csmv$columns
       WHERE conv_rows >0
       AND owner_name =upper('&1')
      );
SPOOL OFF

```

Similarly other necessary scripts can be generated including parfiles for import, scripts to truncate tables, disable/enable triggers, constraints, indices, etc. You must generate them before the next CSSCAN and before truncating the tables. Those activities may change CSMIG owned data and result in inaccurate or incorrect scripts.

Then run the export. It can be run parallel if desired.

```

NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252
exp / parfile=... (use pre-generated parfile)
exp / parfile=...

```

2. Truncate the convertible tables

Before truncating the tables, you may need to disable triggers, foreign key constraints, etc.

Also consider whether or not to drop indices in order to reduce export/import time.

3. CSSCAN again after truncating tables

```

csscan FULL=Y FROMCHAR=WE8MSWIN1252
TOCHAR=AL32UTF8

```

This time there should be only “changeless” left for application data. And the following 3 lines should be listed in the summary scan.txt file:

```

All character type data in the data dictionary remain the
same in the new character set
All character type application data remain the same in the
new character set
The data dictionary can be safely migrated using the CSALTER
script

```

4. Convert to AL32UTF8 using CSALTER

Again, before running CSALTER, be sure to backup your database, shutdown listener and any applications connected to the database, and purge dba_recyclebin. Then run CSALTER as sysdba.

```

SELECT * from nls_database_parameters
WHERE parameter like '%CHARACTERSET%';

shutdown immediate
startup restrict

SPOOL To_AL32UTF8.log
@$ORACLE_HOME/rdbms/admin/csalter.plb
spool off

-- change back job queue, etc to original value

shutdown
startup

SELECT * from nls_database_parameters
WHERE parameter like '%CHARACTERSET%';

```

Your database character set is now AL32UTF8.

5. Re-install datapump packages (see document # 260192.1)

Run scripts at \$ORACLE_HOME/rdbms/admin as sysdba

```

For 10.2.X
    catnodp.sql
    catdph.sql
    catdbp.sql
For 10.1.X
    catnodp.sql
    catdp.sql

```

6. Import the convertible tables

You may use your pre-generated scripts or parfiles to import and can do them parallel.

Again, remember the NLS_LANG should be set to the source character set . In our case:

```

NLS_LANG=AMERICAN_AMERICA.WE8MSWIN1252
imp / parfile=... (use pre-generated parfiles
imp / parfile=...
...

```

Note: It is critical to set the correct NLS_LANG. For both export and import, it should be set to the source character set you export from - the “old” database.

7. Enable or recreate disabled / dropped objects

8. Backup, verify using Unicode client

The best tool for verifying and displaying Unicode data is Oracle's SQL Developer. It is a free GUI tool, a Unicode client, and it is not depending on the NLS_LANG settings.

SQL Developer can be downloaded from Oracle's website:
http://www.oracle.com/technology/products/database/sql_developer/index.html

Another Unicode client is isqlplus.

Now our journey to Unicode has reached its destination – the database and data are AL32UTF8. Congratulations!

SUMMARY

More and more companies are realizing the need for and benefits of using Unicode. However they are often intimidated by its perceived complexity. In the long run, there are just too many circumstances in which Unicode eventually becomes essential, mergers and acquisitions, data and systems consolidation, internationalization of operations, support for new regulations or standards, and just trying to stay ahead of competitors.

Deploying your systems today in Unicode offers many advantages in usability, compatibility, and extensibility. With careful planning and full comprehension of the implementation methods and steps, migration to Unicode can be smooth and successful.

REFERENCES:

- Oracle White Paper: Oracle Unicode database support (May 2005)
- Oracle White Paper: Character Set Migration Best Practices (October 2002)
- Doc 260893.1: Unicode character sets in the Oracle database
- Doc 306411.1: Character Set Consolidation for the Oracle Database
- Doc 333489.1: Choosing a database character set means choosing Unicode
- Doc 788156.1: AL32UTF8 / UTF8 (Unicode) Database Character Set Implications
- Doc 225938.1: Database Character Set Healthcheck
- Doc 158577.1: NLS_LANG Explained (How does Client-Server Character Conversion Work?)

Doc 745809.1: Installing and configuring Cscan in 10g and 11g
(Database Character Set Scanner)
Doc 444701.1: Cscan Output Explained
Doc 276914.1: The National Character Set in Oracle 9i, 10g and 11g
Doc 260192.1: Changing WE8ISO8859P1 / WE8ISO8859P15 or
WE8MSWIN1252 TO (AL32)UTF8
Doc 555823.1: Changing US7ASCII or WE8ISO8859P1 to
WE8MSWIN1252
Doc 341676.1: Difference between WE8MSWIN1252 and
WE8ISO8859P1 character set
Doc 144808.1: Examples and limits of BYTE and CHAR semantics
usage (NLS_LENGTH_SEMANTICS)
Doc 274507.1: Finding out the length semantics of type attributes
Doc 274507.1: Finding out the length semantics of type attributes
Doc 227332.1: NLS considerations in Import/Export - Frequently
Asked Questions
Doc 237593.1: Problems connecting to AL32UTF8 databases from
older versions (8i and lower)