CAMPUSIT TECH NOTE
NOVEMBER 2008

Converting Your Database to Unicode

INTENDED AUDIENCE

- System Administrators
- Database Administrators

OVERVIEW

The purpose of this document is to give the audience a brief explanation of what UNICODE is, its benefits and to direct assist you converting your current database character set to UNICODE.

WHAT IS UNICODE?

Fundamentally unicode is a character set for all the characters and symbols of the world. Prior to the advent of unicode many different character sets and encodings had to be used for different languages. Unicode changed the need for different character sets by assigning a unique number for every character no matter what the platform, no matter what the program and no matter what the language.
See http://www.unicode.org for more...

BENEFITS OF UNICODE

1. It allows for multilingual text using any or all the languages you desire.
2. It ensures interoperability and portability by prescribing conformant behaviour allowing for easier integration with third parties
3. Text in any language can be exchanged worldwide
4. It is internet-ready for use in E-business and E-learning
5. It is continously evolving and extending with support for over 96,000 characters in all modern languages.

CONVERTING YOUR DATABASE TO UNICODE

Oracle has long supported unicode as a character set in its databases. As a matter of course when creating a database instance you can specify what character set you wish to use, unicode is among the many character sets that you can choose from.

However, if you have an existing database and you want to change it to use unicode what do you do?
There are two options:

1. Create a new database instance with unicode specified as the character set, export your existing database and then import it into your new unicode database instance.
2. Use the Oracle initialisation parameter NLS_LENGTH_SEMANTICS to change from byte to character length semantics.

Creating a new database instance, database export and import should be very familiar to you and is not addressed here.

Changing the NLS_LENGTH_SEMANTICS parameter means that all characters in the database are represented by characters be they multi-byte or single byte rather than bytes. This is easily demonstrated in the definition of a table

Characters represented by bytes:
X varchar2(3)

Characters represented by characters:
X varchar2(3 char)

To change an existing schema and its associated data from byte semantics and a single-byte character set to a multi-byte character set such as UTF-8 follow these steps:

1. Export the schema (Quercus for example)

2. Issue an ALTER SYSTEM SET NLS_LENGTH_SEMANTICS=CHAR SCOPE=BOTH command on the target database as the SYS user.
3. Stop and restart the database instance so that the parameter change takes effect.
4. Drop the original schema
5. Recreate the original schema and its tables using IMP command.
6. Import the schema into the target database using the ignore=Y import option.

Note: If your schema includes stored PL/SQL code, you'll need to recompile all those objects in the database. Any object types should be precreated in step 5.
Always run a test conversion before attempting to convert a production system!

Another method of converting to Unicode without using IMP/EXP is by script.
Following steps 1 through 3 above and then use the script below to convert the table definitions:

Connect as the SYS user:

```
set feedback off
set verify off
set serveroutput on
set termout on

exec dbms_output.put_line('Starting build select of columns to be altered');

drop table semantics$;

create table semantics$(s_owner varchar2(40),
s_table_name  varchar2(40),
s_column_name varchar2(40),
s_data_type varchar2(40),
s_char_length number);

insert into semantics$
select C.owner, C.table_name, C.column_name, C.data_type, C.char_length
from   all_tab_columns C, all_tables T
where  C.owner = T.owner
and    T.owner not in ('SYS', 'SYSTEM', 'CTXSYS', 'DBSNMP', 'DMSYS', 'EXFSYS','HR', 'IX', 'MDSYS',
                       'OE', 'OLAPSYS', 'ORDSYS', 'OUTLN', 'SH',   'SYSMAN', 'WKSYS', 'WK_TEST', 'WMSYS',
'XDB')
and    C.table_name = T.table_name
and    C.char_used = 'B'
--  only need to look for tables who are not yet CHAR semantics.
and T.partitioned != 'YES'
-- exclude partitioned tables
and C.table_name not in (select table_name from all_external_tables)
and C.data_type in ('VARCHAR2', 'CHAR')
-- You can exclude or include tables or schemas as you wish, by adjusting
-- "and T.owner not in" as per your requirements
/

Commit;
```

Connect as QUERCUS and drop function based objects

```
Drop index person_upper_surname_idx;
```

Connect as the SYS user:

```
declare
  cursor c1 is select * from semantics$;
  v_statement varchar2(255);
  v_nc number(10);
  v_nt number(10);
begin
  execute immediate
    'select count(*) from semantics$' into v_nc;
  execute immediate
    'select count(distinct s_table_name) from semantics$' into v_nt;
  dbms_output.put_line
    ('ALTERing ' || v_nc || ' columns in ' || v_nt || ' tables');
  for r1 in c1 loop
    v_statement := 'ALTER TABLE ' || r1.s_owner || '.' || r1.s_table_name;
    v_statement := v_statement || ' modify (' || r1.s_column_name || ' ';
    v_statement := v_statement || r1.s_data_type || '(' || r1.s_char_length;
    v_statement := v_statement || ' CHAR))';
    execute immediate v_statement;
  end loop;
  dbms_output.put_line('Done');
end;
/

Commit;
```

Connect as QUERCUS and recreate function based objects

```
create index PERSON_UPPER_SURNAME_IDX on PERSON (UPPER(SURNAME)) tablespace QUERCUS_IDX;
```

Note: If your schema includes stored PL/SQL code, you'll need to recompile all those objects in the database.
Always run a test conversion before attempting to convert a production system!

REFERENCES AND USEFULL INFORMATION

Oracle Magazine – Globalize with Character Semantics