

Evitez les erreurs de conversion grâce aux NLS



par Fabien Celaia ([fadace](#))

Date de publication : 29.07.2006

Dernière mise à jour : 15.02.2009

Evitez de pourrir la qualité de vos données en saisissant la notion de conversion de tables de caractères en mode client serveur

I - Introduction.....	3
II - Choix du jeu de caractères au niveau serveur.....	3
III - Cas d'exemple.....	4
IV - La problématique.....	4
V - Explication.....	5
VI - Résolution.....	7
VII - Modification du jeu de caractères d'une base existante.....	8
VIII - Derniers pièges à éviter.....	8
VIII-A - JDBC.....	8
VIII-B - Editeur graphique ou texte.....	9
VIII-C - Polices de caractères.....	9
IX - Documentation.....	9

I - Introduction

Oracle, comme la plupart des SGBDR client-serveur, est en mesure de gérer divers formats de paramètres locaux, tels le format de la date, celui de l'heure, des jours de la semaine, du format de monnaie, le jeu de caractères, etc.

Plusieurs personnes (clients) n'ayant pas les mêmes paramètres peuvent ainsi faire cohabiter sur une même base (serveur) leurs données.

Chacune de ces valeurs est spécifiée au niveau du serveur comme nous le verrons plus bas, mais peut l'être au niveau d'une base de registre ou de variables d'environnements au niveau du client.



Concepts de base : au moment où les valeurs diffèrent entre client et serveur, Oracle se met à convertir.

Tout devrait donc aller pour le mieux dans le meilleur des mondes. Malheureusement, la méconnaissance des paramètres influençant ces configurations peut générer un remède pire que le mal : des erreurs de conversions irrécupérables car multiples et diverses.

Nul besoin d'être en environnement international pour se trouver confronté à ce genre de problème : Windows, la plateforme cliente la plus répandue, nous offre un cas d'école. En effet, ce système d'exploitation gère à lui seul au moins deux genres de pages de caractères : le ANSI pour sa partie graphique, et le ASCII pour sa partie mode texte (session Dos, cmd ou command, c'est selon.)

Dès lors, comment Oracle s'y retrouve-t-il ? Et bien par défaut, force est de constater qu'il ne s'y retrouve pas très bien !!! En voici un exemple probant.

En fait, nombre de DBA renseignent incorrectement NLS_LANG sur leurs clients en la faisant coïncider avec le code de la base. Le résultat - double erreur de conversion s'annulant - est traître. Tout semble correct à première vue, mais lorsqu'une application externe arrive ou lorsqu'une session gère correctement la conversion, c'est le souk au niveau de vos données... Avoir un client et un serveur avec le même NLS_LANG empêche une quelconque conversion par Oracle. Oracle stocke donc de mauvais caractères dans ses tables mais cela ne se voit pas dans l'application fautive, et les performances s'en retrouvent améliorées puisqu'aucune conversion n'est faite.

II - Choix du jeu de caractères au niveau serveur

Bien des administrateurs ont fait de mauvais choix en imposant le jeu de caractères du système d'exploitation, en invoquant l'excuse que ce choix éviterait des conversions coûteuses en performance.

Ce choix est erroné : les conversions se jouent entre client et base de données, et pas entre système d'exploitation hébergeant le serveur de données et ce dernier. Il est vrai cependant que souvent les scripts de batch sont exécutés sur la machine même, celle-ci jouant alors les deux rôles : client et serveur.

Voici donc quelques règles régissant le choix correct du jeu de caractères au niveau de la base de données.

- 1 Dans le cas où l'application cliente principale fonctionne sous Windows, optez pour le jeu de caractères Windows sur la base
- 2 Dans le cas où l'application cliente est une application Java, optez pour l'UTF
- 3 Dans le cas où plusieurs applications de types divers cohabitent, optez pour le jeu de caractères le plus étendu
- 4 Dans le cas de purs traitements batchs lancés sur la machine hébergeant le serveur ou ayant même système d'exploitation, et dans le cas où vos traitements ne se font pas via java (sinon retour à la règle 2), optez pour le jeu de caractères similaire au système d'exploitation

Oracle recommande de plus en plus souvent l'utilisation de l'UTF8, car il s'agit d'un superset de caractères (incluant la plupart des autres). Le coût à payer en est la place de stockage et quelques problèmes liés aux tailles des champs plus difficiles à déterminer.

III - Cas d'exemple

Imaginons un Serveur Oracle installé sur un système Sun Solaris, dont le jeu de caractère a été déterminé par le DBA en européen comprenant le signe de l'Euro. De plus, tous les clients sont sous Windows, mais certaines applications lancent des traitements batch.

En résumé, nous avons donc, sans trop nous en rendre compte:

- Un système d'exploitation utilisant du ISO1
- Un SGBDR utilisant un ISO modifiée (P15)
- Un client graphique utilisant de l'ANSI
- Des traitements batch lancés en mode Dos, utilisant donc du CP850

Traduisons cela en "Oracle"

- Un système d'exploitation utilisant du AMERICAN_AMERICA.WE8ISO8859P1
- Un SGBDR utilisant un ISO1 modifié AMERICAN_AMERICA.WE8ISO8859P15
- Un client graphique utilisant de l'ANSI FRENCH_FRANCE.WE8MSWIN1252
- Des traitement batch lancés en mode Dos, utilisant donc du FRENCH_FRANCE.WE8PC850

Comme nous l'avons vu plus haut, le décodage est aisé : langue, territoire et jeu de caractères.

IV - La problématique

Créons-nous une table simple ne comportant qu'un champ "chaîne de caractères" et laissons tous les paramètres par défaut que l'Install Oracle nous a mis.

```
CREATE TABLE TESTCONVERSION
  (V VARCHAR2(50)) ;
COMMIT ;
```

Insérons maintenant une ligne via chaque client:

Via SQLPlusw (mode Windows graphique):

```
SQL> INSERT INTO TESTCONVERSION VALUES('Test éphémère à Windows graphique.');
```

```
COMMIT ;
```

Via SQLPlus (mode Dos):

```
SQL> INSERT INTO TESTCONVERSION VALUES('Test éphémère à Windows texte.');
```

```
COMMIT ;
```

Via SQLPlus sous Unix:

```
SQL> INSERT INTO TESTCONVERSION VALUES('Test éphémère à Unix.');
```

```
COMMIT ;
```

Affichons ensuite le résultat de notre gabegie:

Sous SQLPlusw de Windows

```
SQL> SELECT * FROM TESTCONVERSION ;

V
-----
Test éphémère à Windows graphique.
Test ¿ph¿m¿re ¿ Windows texte.
Test ¿ph¿m¿re ¿ Unix.
```

Sous SQLPlus sous ligne de commande Dos

```
SQL> SELECT * FROM TESTCONVERSION ;

V
-----
Test ÚphÚmÚre Ó Windows graphique.
Test +ph+mère + Windows texte.
Test +ph+m+re + Unix.
```

Sous SQLPlus sous ligne de commande Unix

```
SQL> SELECT * FROM TESTCONVERSION ;

V
-----
Test ephemere a Windows graphique.
Test ?ph?mSre ? Windows texte.
Test ?ph?m?re ? Unix.
```

Comme vous le voyez, rien ne va plus ! ... et même la conversion de mes copier/coller pour cet article me changent les caractères problématiques... Nous sommes face d'un sérieux problème quant à la qualité de nos données.

V - Explication

En fait, que se passe-t-il réellement ? Et bien Oracle tente de traduire tant bien que mal les caractères accentués d'un code page à l'autre... en se basant sur de fausses indications. Comme vous le notez, il est incapable non seulement de rendre correctement les données des autres, mais aussi les données propres à chaque session... hormis celle sous Windows graphique...

Une explication s'impose...

Avant tout, déterminons avec certitude les paramètres attribués à l'instance Oracle, soit au niveau serveur.

```
SQL> select * from NLS_DATABASE_PARAMETERS ;
PARAMETER                                VALUE
-----
NLS_LANGUAGE                             AMERICAN
NLS_TERRITORY                             AMERICA
NLS_CURRENCY                               $
NLS_ISO_CURRENCY                           AMERICA
NLS_NUMERIC_CHARACTERS                    .,
NLS_CHARACTERSET                           WE8ISO8859P15
NLS_CALENDAR                               GREGORIAN
NLS_DATE_FORMAT                            DD-MON-RR
NLS_DATE_LANGUAGE                           AMERICAN
NLS_SORT                                   BINARY
NLS_TIME_FORMAT                             HH.MI.SSXF AM
NLS_TIMESTAMP_FORMAT                       DD-MON-RR HH.MI.SSXF AM
NLS_TIME_TZ_FORMAT                          HH.MI.SSXF AM TZR
NLS_TIMESTAMP_TZ_FORMAT                    DD-MON-RR HH.MI.SSXF AM TZR
NLS_DUAL_CURRENCY                           $
NLS_COMP                                   BINARY
```

```
NLS_LENGTH_SEMANTICS      BYTE
NLS_NCHAR_CONV_EXCP      FALSE
NLS_NCHAR_CHARACTERSET    AL16UTF16
NLS_RDBMS_VERSION         10.2.0.1.0
```

20 ligne(s) sélectionnée(s).

Par concaténation des lignes NLS_LANGUAGE, NLS_TERRITORY et NLS_CHARACTERSETS, nous obtenons donc AMERICAN_AMERICA.WE8ISO8859P15. Soit en clair:

- La langue anglaise (AMERICAN)
- Des codes locaux américains (pour le format des dates, des monnaies) (AMERICA)
- Un jeu de caractère ISO (ISO8859) pour l'Europe de l'Ouest (WE) codé sur 8 octets, avec une spécificité (code page 15, pour intégrer le signe de l'Euro).

Au niveau de notre session, il est toujours possible de voir quels sont les paramètres locaux via

```
SQL> SELECT * FROM NLS_SESSION_PARAMETERS ;

PARAMETER                                VALUE
-----
NLS_LANGUAGE                             FRENCH
NLS_TERRITORY                             FRANCE
NLS_CURRENCY                              □
NLS_ISO_CURRENCY                          FRANCE
NLS_NUMERIC_CHARACTERS                    ,
NLS_CALENDAR                              GREGORIAN
NLS_DATE_FORMAT                           DD.MM.RRRR
NLS_DATE_LANGUAGE                          FRENCH
NLS_SORT                                   FRENCH
NLS_TIME_FORMAT                            HH24:MI:SSXFF
NLS_TIMESTAMP_FORMAT                       DD/MM/RR HH24:MI:SSXFF
NLS_TIME_TZ_FORMAT                         HH24:MI:SSXFF TZR
NLS_TIMESTAMP_TZ_FORMAT                    DD/MM/RR HH24:MI:SSXFF TZR
NLS_DUAL_CURRENCY                          □
NLS_COMP                                   BINARY
NLS_LENGTH_SEMANTICS                       BYTE
NLS_NCHAR_CONV_EXCP                        FALSE
```

17 ligne(s) sélectionnée(s).

Si les variables NLS_DATE_LANGUAGE ou NLS_SORT ne sont pas définies, elles sont issues par défaut du NLS_LANGUAGE.

Si les variables NLS_CURRENCY, NLS_DUAL_CURRENCY, NLS_ISO_CURRENCY, NLS_DATE_FORMAT, NLS_TIMESTAMP_FORMAT, NLS_TIMESTAMP_TZ_FORMAT, NLS_NUMERIC_CHARACTERS ne sont pas définies, elles sont issues par défaut du NLS_TERRITORY.

Pour SQLPlusw, voici ce que Oracle fait:

- 1 Il sait que le serveur est en WE8ISO8859P15
- 2 Il sait que le client est en WE8MSWIN1252
- 3 Il traduit ce qu'il reçoit du client (WE8MSWIN1252) en WE8ISO8859P15 et l'insère dans la table correctement
- 4 Lors du SELECT, il traduit ce qu'il a du serveur (WE8ISO8859P15) pour le client (WE8MSWIN1252)
- 5 Les données apparaissent correctement à l'écran.

Pour SQLPlus sous Dos, voici ce que Oracle fait:

- 1 Il sait que le serveur est en WE8ISO8859P15
- 2 Il croit que le client est en WE8MSWIN1252 : c'est ce que lui dit la base de registre Windows, mais elle a tort dans ce cas puisqu'il s'agit en fait du WE8PC850
- 3 Il traduit ce qu'il reçoit du client (WE8PC850) en utilisant le faux convertisseur (WE8MSWIN1252) en WE8ISO8859P15 et l'insère dans la table incorrectement

- 4 Lors du SELECT, il traduit ce qu'il a du serveur (WE8ISO8859P15, mais déjà corrompu par la conversion lors de l'insert) pour le client (WE8MSWIN1252)
- 5 Les données apparaissent incorrectement à l'écran (double conversion en erreur).

VI - Résolution


Il suffit de paramétrer correctement la valeur NLS_LANG sur chacun des clients. Cette variable doit refléter la page code utilisée par le client, et pas - comme certain le pensent - la page code configurée au niveau de la base Oracle. Dans le cas où l'on se retrouve avec un NLS_LANG identique au choix fait pour la base Oracle, la conversion ne se fait alors pas !

Pour Windows, c'est dans la base de registre que cela se joue.

```
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\KEY_O10R2\NLS_LANG=FRENCH_FRANCE.WE8MSWIN1252
```

Pour une session Dos, c'est la variable d'environnement qui peut être appliquée.

```
set NLS_LANG=FRENCH_FRANCE.WE8PC850
```

 *La tentation est grande, mais veillez surtout à ne JAMAIS mettre cette variable d'environnement dans la base de registre : elle prendrait le pas sur la valeur du registre ORACLE WE8MSWIN1252 !*

même méthode pour Unix, à insérer dans le .profile ou .login. Attention, selon les systèmes, la valeur peut être différente.

```
export NLS_LANG=AMERICAN_AMERICA.WE8ISO8851P1
```

Après configuration, toutes vos insertions passeront correctement et tous vos SELECT vous retourneront les lignes correctes.

Sous SQLPlus et SQLPlusw, quel que soit le système d'exploitation

```
SQL> SELECT * FROM TESTCONVERSION ;

V
-----
Test éphémère à Windows graphique.
Test éphémère à Windows texte.
Test éphémère à Unix.
```

Pour les NLS apparaissant dans la NLS_SESSION_PARAMETERS, il est possible d'influer au niveau des variables d'environnement comme nous l'avons vu plus haut, mais aussi au niveau session, ce qui est parfois préférable plutôt que de risquer d'impacter des applications existantes.

```
SQL> SELECT VALUE FROM NLS_SESSION_PARAMETERS WHERE PARAMETER = 'NLS_DATE_FORMAT' ;

VALUE
-----
DD-MON-RR

1 ligne(s) sélectionnée(s).

SQL> SELECT SYSDATE FROM DUAL ;

SYSDATE
-----
28-JUIL.-06
```

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD.MM.YYYY'


Session modifiée.

SQL> SELECT SYSDATE FROM DUAL ;

SYSDATE
-----
28.07.2006
```

VII - Modification du jeu de caractères d'une base existante

Le choix du jeu de caractères de la base de données est fait à la création de ladite base. Sa modification a été longtemps considérée comme impossible : il fallait passer par un export/import coûteux

Il existe en fait une méthode qui permet, sous quelques conditions, de modifier le jeu de caractères courant. Elle est documentée dans Oracle Metalink sous la  [Doc-ID 225912.1](#).

Elle se résume en quelques points :

- 1 S'assurer que le jeu de caractères choisi englobe tous les caractères utilisés dans votre base, via l'utilisation du scanner *csscan*
- 2 Utiliser la commande ALTER DATABASE CHARACTER SET jusqu'à la v.9, et CSALTER dès la version 10
- 3 Rejouer un *csscan* pour s'assurer qu'aucune corruption n'est avérée
- 4 Dans le cas où un problème persisterait, utiliser les commandes EXP et IMP, mais pas le datapump.

VIII - Derniers pièges à éviter

VIII-A - JDBC

Oracle permet une connexion JDBC de 2 types distincts

- JDBC **thick** (lourd) : la couche JDBC s'implémente au dessus de la couche cliente Oracle.
- JDBC **thin** (léger) : la connexion se fait directement via JDBC.

Le JDBC thin n'est donc pas réactif aux variables d'environnement NLS. Or tout ce qui est JDBC est configuré par défaut pour utiliser le AMERICAN_AMERICA... Vos dates vont donc se retrouver dans un triste état...

Il est possible de remédier à cela en exécutant dans la session les 2 ordres

```
ALTER SESSION SET NLS_LANGUAGE='FRENCH' ;
ALTER SESSION SET NLS_TERRITORY='FRANCE' ;
```

Depuis la version 10.2 d'Oracle, il est possible de le faire via un trigger sur connexion:

```
CREATE OR REPLACE TRIGGER NLS_CONFIG_TRG
AFTER LOGON ON DATABASE
BEGIN
execute immediate 'ALTER SESSION SET NLS_LANGUAGE="FRENCH"';
execute immediate 'ALTER SESSION SET NLS_TERRITORY="FRANCE"';

-- et pour palier à l'année sur 2 digits
execute immediate 'ALTER SESSION SET NLS_DATE_FORMAT="DD.MM.RRRR"';
execute immediate 'ALTER SESSION SET NLS_TIMESTAMP_FORMAT="DD.MM.RRRR HH24:MI:SSXFF"';
END;
/
```

Pour compliquer encore un peu, il faut savoir que la plupart des applications Java utilisent un jeu de caractère UNICODE. Si donc le jeu de caractères de votre base de données n'est pas défini en Unicode (UTF), une conversion doit s'opérer.

Jeu de caractères de la base Oracle	Conversion effectuée depuis le client Java
US7ASCII WE8DEC WE8ISO8851P1 UTF8	conversion effectuée graces aux classes de base classes**.zip, classes**.jar
WE8ISO8851P15 WEMSWIN1252 EE8MSWIN1250	conversion effectuée graces aux classes livrées avec le JDBC d'Oracle nls_charset**.zip, nls_charset**.jar

Les ** sont a remplacer par {11, 12, 13, 14} selon la version JDK utilisée {1.1, 1.2, 1.3, 1.4}

VIII-B - Editeur graphique ou texte

Si vous avez configuré correctement votre session Dos en CP850, veillez à ne pas lancer via SQLPlus un fichier de script qui aurait été écrit avec Notepad ou Wordpad.

Si tel a été le cas, vous pouvez forcer le codage dans votre script avec un

```
ALTER SESSION SET NLS_LANGUAGE=FRENCH_FRANCE.WE8MSWIN1252;
```

VIII-C - Polices de caractères

Il se peut que, malgré une configuration correcte de NLS_LANG, certains caractères n'apparaissent pas correctement. c'est le cas par exemple au travers de SQLPlusW qui n'affiche pas correctement le caractère □. La cause en est simple : SQLPlusW utilise une police de caractère "Courier" qui n'a pas de □ de sa page. Votre base est correcte, votre client aussi, mais votre affichage laisse alors à désirer !

IX - Documentation

- Oracle Metalink Note 115001.1 : NLS_LANG Client Settings and JDBC Drivers
- Oracle Metalink Note 251044.1 : How to set a NLS session parameter at database level for all sessions
- Oracle Metalink Note 225912.1 : Changing the Database Character Set - a short overview
- Oracle Metalink Note 158577.1 : NLS_LANG Explained (How dows Client-Server Character Conversion Work?)
- Oracle Metalink Note 179133.1 : The correct NLS_LANG in a Windows Environment
- Oracle Metalink Note 241047.1 : The Priority of NLS Parameters Explained