generix collaborative generation

generix

[Edition 1.03 du 22/11/2005]

Guide de mise en œuvre : Oracle Text

Table des matières

1 Introduction

Ce document doit accompagner toute prestation d'installation Oracle Text.

1.1 Pré-requis technique

L'utilisation d'Oracle Text avec eGx nécessite des versions précises d'Oracle pour faire fonctionner l'ensemble des fonctionnalités implémentées.

- 9.2.0.6 (9iR2) pour la version GCE 1.0
- 10.1.0.4 (DB 10g R1) pour la version GCE 1.1

Release Oracle SGBDR

Des versions précédentes ont été testées, elles comportent des bugs d'importance sur ce module d'Oracle. **C'est donc un pré-requis fondamental.**

Ce module d'Oracle n'est pas une option payante : les scripts de création d'objets dédiés aux recherches Oracle Text étant livrés en Open Source, la maintenance GENERIX ne sera possible que si l'exploitation d'Oracle Text reste dans le standard de la livraison.

Oracle Text et UTF8

L'option Oracle Text n'est actuellement pas supportée par Oracle sur des bases en UTF8.

1.2 Principes appliqués à eGx

1.2.1 Concept d'Oracle Text

OT (Oracle Text) permet de réaliser des recherches « déstructurées » dans des tables spécifiquement dédiées à cela. Le moteur de recherche (syntaxiquement parlant) est très semblable à celui utilisé par l'archivage XDB. Les grands principes de recherches sont équivalents à ceux que l'on retrouve sur les moteurs de recherches Internet habituels (Google, Yahoo …).

Il existe 3 grands types de recherches avec **OT** :

- Plain text
- HTML
- XML

Pour le cas qui nous intéresse, les recherches de type Plain Text vont permettent de retrouver des informations dans des **tables spécifiques** avec une syntaxe très simple en regard d'une syntaxe habituelle de recherche multi-critères (en l'occurrence nous n'allons utiliser qu'un seul critère sur un seul champ !).
L'implémentation dans eGx est donc relativement aisée dès lors que l'on maitrise le fichier de configuration.

En aucun cas, ce type de recherche ne vient se substituer à une recherche classique, elle vient la compléter. C'est un plus et une facilité pour l'utilisateur final du produit.

L'implémentation autour d'eGx se situe sur la recherche produit et la recherche client. En effet, suivant les contextes, ces tables peuvent être très volumineuses et on peut souhaiter faire des recherches aussi bien sur des informations de la table principal que sur des informations des tables secondaires (au sens du MCD **GENERIX** du terme). **Ces recherches doivent principalement s'appliquer sur des recherches typées « libellés, noms … ».**

Par exemple, **OT** va nous permettre dans la même requête de chercher une information aussi bien sur les libellés produits que sur les textes libres associés. De plus on peut paramétrer **OT** pour que la recherche soit indifférente aux accents, à la casse, aux séparateurs de mots … Une recherche avec les valeurs suivantes donnera le même résultat :

- **BEBE**
- **bebe**
- **Bébé**

D'autre part il est possible d'exploiter une syntaxe plus évoluée (à peu près identique à celle de Google) :

- **Bébé | (pot chambre)**

Le symbole « | » (pipe) est utilisé pour appliquer un « ou » logique entre les 2 opérandes. Les parenthèses permettent de regrouper les opérandes entre 2 opérateurs. L'espace et le caractère « & » correspondent à un opérateur « et » logique et sont strictement similaires.

Important : le caractère « * » remplace le caractère « % » dans les recherches partielles. Par exemple :

- Béb* pot
  ➔ recherche sur tous les libellés contenant le début de mot **« Béb »** ET contenant le mot **« pot »**

Le fonctionnement est basé sur des indexes spécifiques à **OT** : stockage de chacun des mots constituant le champ de recherche dans le format spécifié par le contexte d'index.
➔ Index de type **DOMAIN**

1.2.2 Nouvelles tables

Deux nouvelles tables vont permettent de stocker les informations au format **OT**, elles sont constituées du nom de la table de base suffixées par _**REC** (RECherche) :

- PRO_REC
- TIE_REC

Les champs contenus dans ces tables sont systématiquement :

- La clé primaire de la table qui doit être définie de la même façon que pour la table de base
- Un champ qui permettra de stocker toutes les informations sur lesquelles on souhaite faire la recherche : champ « *TEXTE* »

Recherche « plain text » sur la table PRO ➔ PRO_REC

- CODSOC : type integer
- CODPRO : type VARCHAR2(16)
- TEXTE : type VARCHAR2(2000)

Recherche « plain text » sur la table TIE ➔ TIE_REC

- CODSOC : type integer
- TYPTIE : type VARCHAR2(3)
- SIGTIE : type VARCHAR2(12)
- TEXTE : type VARCHAR2(4000)

Arbitrairement, le stockage sur le champ texte ne doit pas dépasser 2000 caractères pour les produits. Suivant le contexte client, on pourra être amené à l'augmenter à une valeur supérieure

(dans la limite de **4000 caractères** fixée par Oracle, on ne peut utiliser pour nos recherches des types comme les LOBs …).

Chacune de ces tables doit avoir sa propre clé primaire de la même manière que la table « standard ». Par exemple, pour la table PRO la clé primaire est définie pour les champs (CODSOC,CODPRO).

Le champ « *TEXTE* » sera la concaténation séparée par des espaces des champs des différentes tables sur lesquelles on souhaite faire la recherche. L'alimentation de ce champ doit donc être réalisé dans la limite de sa définition en longueur (donc 2000 en standard)
➔ des procédures en PL/SQL vont nous permettre d'alimenter les tables dédiées à ces recherches

Alimentation des tables de recherche

Il serait possible d'alimenter les tables de recherche par d'autre moyens (Java, SQL*Loader …). Le PL/SQL a l'avantage d'être très connu et facilement adaptable sur site.

1.3 Création des objets Générix pour Oracle Text

Pré requis Oracle

Avant de passer les scripts de création d'objets, il est indispensable d'attribuer le rôle CTXAPP au compte GENERIX (en général SOC1) depuis un compte DBA ou SYSDBA.
Par exemple sous SQL*Plus :

SQL> connect system/manager
SQL> grant ctxapp to soc1 ;

Le rôle CTXAPP est créé lors de l'installation d'**OT** depuis le DBCA.

Le script ***ORA_TEXT_PACK.SQL*** permet de créer les tables et indexes spécifiques (+ les contextes d'indexes), ainsi que certains triggers qui permettront une alimentation au fil de l'eau.

On va distinguer 2 cas d'utilisation des tables **OT** :

- Mise à jour en masse de toute la table
- Mise à jour différentielle : seules les données modifiées seront mises à jour

Afin de faciliter la maintenance et de regrouper les différentes fonctionnalités, les procédures de mise à jour sont fournies sous la forme d'un package Oracle nommé **GXP_OT**

On notera qu'au même titre que les tables habituelles, les statistiques Oracle doivent être calculées sur la table.

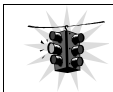
Création des objets dans le schéma GENERIX (habituellement SOC1)

Sqlplus <SOCx>/<INFORx>
SQL> *@ora_text_pack.sql*

(le script *ora_text_pack.sql* fourni par Generix sur le kit de paramétrages Generix)

Tableau 1 : objets contenus dans le script de création

Nom	Type	Commentaire
PRO_REC	Table	
PRO_REC_MOD_SEQ	Séquence	Séquence pour la table PRO_REC_MOD
PRO_REC_MOD	Table	Table permettant de stocker les codes produits modifiés pour le mode de mise à jour différentiel
PRO_REC_INDEX_SET	Index domaine	
PRO_REC_STORAGE	Index stockage	Paramètres de stockage physique de l'index de domaine
PRO_REC_LEXER	Index lexique	Définition des paramètres du lexique à utiliser pour le stockage de l'index (langue, séparateurs …)
PRO_REC_INDEX	Index	Index de type domaine basé sur les définitions données ci-dessus
PRO_REC_IDX1	Contrainte	Contrainte primaire sur les champs (CODSOC,CODPRO)
TW_PRO_REC_MOD	Trigger	Trigger permettant d'alimenter la table PRO_REC_MOD, il est désactivé par défaut
TIE_REC	Table	
TIE_REC_MOD_SEQ	Séquence	Séquence pour la table TIE_REC_MOD
TIE_REC_MOD	Table	Table permettant de stocker les tiers modifiés pour le mode de mise à jour différentiel
TIE_REC_INDEX_SET	Index domaine	
TIE_REC_STORAGE	Index stockage	Paramètres de stockage physique de l'index de domaine
TIE_REC_LEXER	Index lexique	Définition des paramètres du lexique à utiliser pour le stockage de l'index (langue, séparateurs …)
TIE_REC_INDEX	Index	Index de type domaine basé sur les définitions données ci-dessus
TIE_REC_IDX1	Contrainte	Contrainte primaire sur les champs (CODSOC,TYPTIE,SIGTIE)
TW_PRO_REC_MOD	Trigger	Trigger permettant d'alimenter la table TIE_REC_MOD, il est désactivé par défaut
GXP_OT	Package	Package permettant la gestion de l'alimentation des tables PRO_REC et TIE_REC



Attention : Suppression avant création !

Le script commence systématiquement par supprimer les objets (PRO_REC …) avant de les créer. La première exécution de ce script générera donc quelques erreurs dont il ne faut pas tenir compte.

D'autre part, le script ne provoque pas l'alimentation des tables lors de son exécution. Ne pouvant pas préjuger de la volumétrie à traiter, il est indispensable que ce soit une action « manuelle ». Ceci fait l'objet d'un chapitre à part dans le présent document.

1.4 Le package GXP_OT

Détail des fonctions et procédures stockées disponibles dans le package :

- **deleteRepetitive**
- **PRO_REC_COMPLETE** : mise
- **PRO_REC_FAST**
- **PRO_REC_UNIQUE** : fonc
- **TIE_REC_COMPLETE** : mise
- **TIE _REC_FAST**
- **TIE _REC_UNIQUE** : fonc

Multi-entité

Le mode de fonctionnement de la recherche implique que les tables « filles » doivent se trouver sur le même code société que la table « maître ». Par exemple si la tables des produits est partagées sur la société 1 (pro.codsoc=1), les gencod et les textes libres associés doivent eux-aussi se trouver sur la société 1. Il s'agit du mode de fonctionnement standard de GENERIX. Si ces tables ne sont pas partagées, la problématique ne se pose pas non plus.

1.5 Mise à jour pour un seul produit

Exemple de mise à jour pour un seul produit (sous SQL*Plus) :

```
rem Mise en place de la taille du buffer pour sortie écran (CR)
set serveroutput on size 1000000
declare
  x boolean;
begin
  x := gxp_ot.pro_rec_unique(1, 'BR101');
end;
/
```

Exemple de mise à jour pour un seul client (sous SQL*Plus) :

```
rem Mise en place de la taille du buffer pour sortie écran (CR)
set serveroutput on size 1000000
declare
  x boolean;
begin
  x := gxp_ot.tie_rec_unique(1, 'CLI', 'BRCLI101');
end;
/
```

1.6 Mise à jour de masse : exemple produits

Suivant la fréquence de mise à jour des tables concernées, on peut prévoir une mise à jour quotidienne. Il suffit pour cela d'exécuter la procédure **PRO_REC_COMPLETE** du package précité. La mise à jour est complète, c'est-à-dire que la table PRO_REC est d'abord vidée avant d'être mise à jour à partir de tous les produits de la table PRO.

Par exemple, sous SQL*Plus :

```
rem Mise en place de la taille du buffer pour sortie écran (CR)
set serveroutput on size 1000000
begin
  gxp_ot.pro_rec_complete ;
end ;
/
```

Remarque

Validation de transaction : COMPLETE

En mode complet, les créations dans la table PRO_REC sont validées par paquet. Le commit est effectué toutes les « nbc » lignes (variable locale à la fonction).

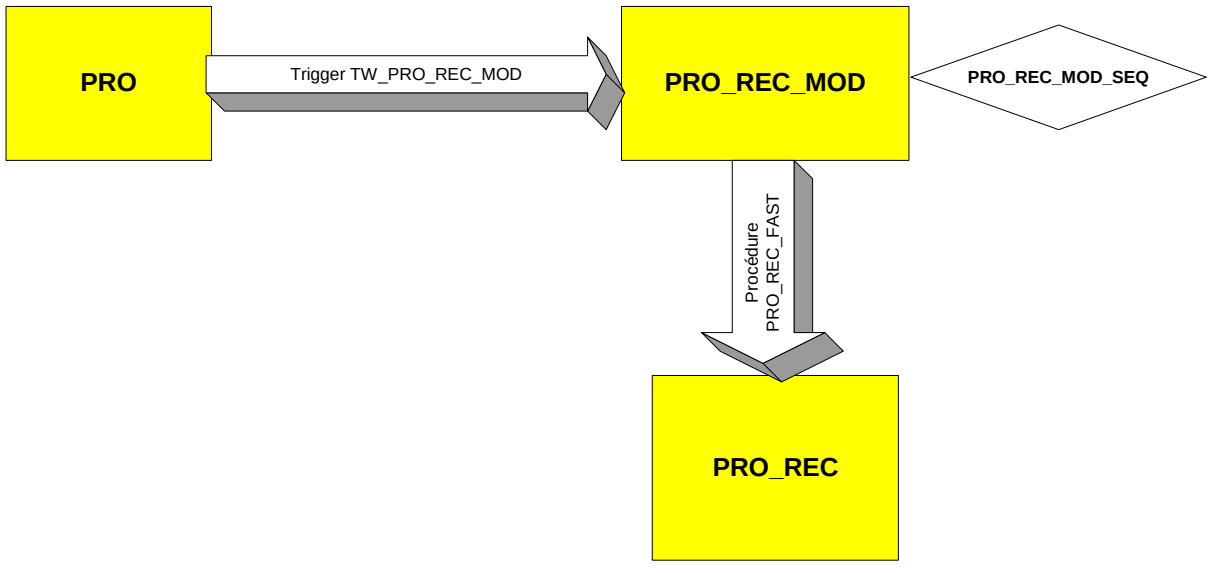
La table PRO_REC est mise à jour avec les libellés de la fiche produit, les Gencod et les textes libres associés. Les traductions ne sont pas prises en compte, ceci peut faire l'objet d'un développement spécifique en projet.

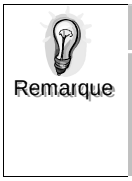
Le mode de fonctionnement est strictement équivalent pour la mise à jour des tiers.

1.7 Mise à jour différentielle : exemple produits

On va réaliser dans ce cas une mise à jour différentielle par rapport à la liste des produits que l'on va trouver dans la table PRO_REC_MOD. Cette table est alimentée par le trigger TW_PRO_REC_MOD.

Schéma :





Validation de transaction : FAST

En mode différentiel, les créations dans la table PRO_REC sont validées au global à la fin de la mise à jour. La volumétrie à traiter est normalement beaucoup plus faible que lors d'une mise à jour complète.

La séquence PRO_REC_MOD_SEQ permet d'ordonnancer l'ordre dans lequel le traitement de mise à jour doit traiter les transactions.

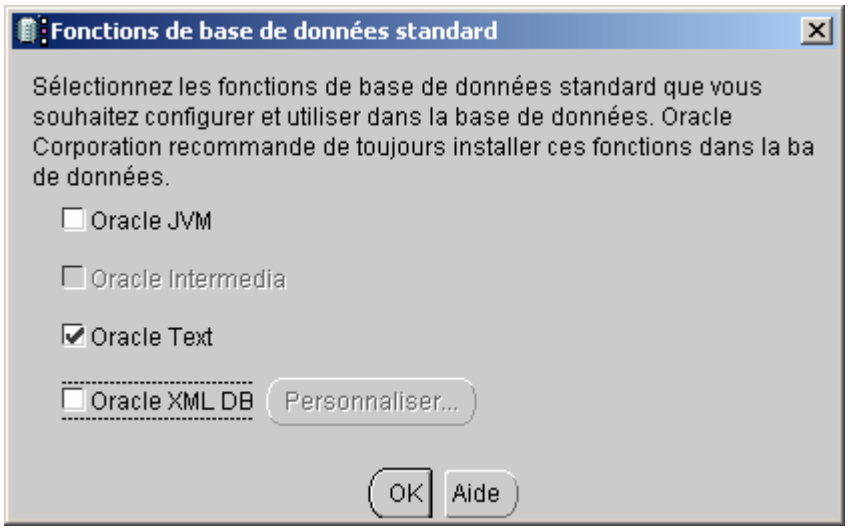
Le mode de fonctionnement est strictement équivalent pour la mise à jour des tiers.

2 Mise en œuvre de l'option Oracle Text

Remarque	**Installation des packages Oracle Text**
	Il est possible d'installer les packages … sans passer par le DBCA, néanmoins ces manipulations exigent une très bonne connaissance du module concerné dans la version d'Oracle. Ce mode d'installation est **TRES FORTEMENT DECONSEILLE !**

2.1 Création d'une nouvelle instance Oracle pour Generix

Dans le cadre d'une première installation de Generix ou de eGx, il est nécessaire de créer une instance Oracle pour Generix. L'option *Oracle Text* doit être validée dès la création de l'instance :



Cela a pour effet d'installer de nouveaux tablespaces et de nouveaux comptes Oracle pour gérer les objets Oracle Text.

Nota bene	**Pour plus d'informations …**
	se référer aux documentations Oracle suivantes : Oracle Text Application Developer's Guide : a96517 Oracle Text Reference :a96518

Nous vous **recommandons fortement** d'utiliser les modèles DBCA ainsi que les scriptes SQL fournis par Generix (présents sur le kit de paramétrages Generix). Ces derniers valident automatiques toutes les options Oracle obligatoires pour un bon fonctionnement de Generix :

La procédure de création d'une instance Oracle pour Generix est la suivante :

1. Créer de la base Oracle partir de DBCA et des modèles fournis par Genérix :
   (Se reporter au Guide de référence technique)

   Les étapes effectuées sont les suivantes :

   - Création de l'instance Generix
   - Création du schéma « SOC1 »

REMARQUE : A partir des kits d'installation de generix V5.2-00 et des kits de paramétrage GCE1.1 du 23/11/2005, le schéma SOC1 est créé avec le rôle CTXAPP.

2. Si nécessaire, créer les autres sociétés physiques SOCx avec le rôle CTXAPP
   (Se reporter au Guide de référence technique)

3. Sur **tous** les schémas de Générix,

   - Créer les objets Generix à partir du script *SCHEMA_O.SQL* et du répertoire *SQL_O*
     (Se reporter au Guide de référence technique)

   - Créer les objets Generix pour Oracle Text à partir du script *ORA_TEXT_PACK.SQL*
     (Se reporter au Guide de référence technique)

   - Importer les données dans la base (dump ou str)
     (Se reporter au Guide de référence technique)

   - Effectuer une première alimentation des tables OT :

     ```
     Sqlplus <SOCx>/<INFORx
     SQL> exec gxp_ot.tie_rec_complete
     SQL> exec gxp_ot.pro_rec_complete
     ```

   Dans le cas ou l'on ne souhaite pas faire de recherches OT sur les tiers par exemple, seule la 2ième ligne est exécuter bien entendu.

   - Mettre en place une stratégie de rafraîchissement complet périodique.
     (voir chapitre 4 de ce documment)

2.2 Mise à jour de la version de Oracle

Dans le cas d'une montée de version de Oracle (sans upgrade de Generix), la procédure est la suivante, elle doit être appliquée **sur toutes les instances** :

1. Rechercher **tous les schémas** présents sur cette instance :

Exemple :

```
Sqlplus system/manager
select username from dba_users where username like 'SOC%';

>  USERNAME
>  --------
>  SOC9800
>  SOC1
>  SOC8000
>  SOC9700
```

2. Rechercher la structure des différents tablespaces DATA et INDEX

Exemple :

```
Sqlplus system/manager
select tablespace_name,status,contents,extent_management from dba_tablespaces;

TABLESPACE_NAME   STATUS    CONTENTS  EXTENT_MAN
--------------------------------  ---------      ---- ----------
SYSTEM                 ONLINE    PERMANENT   LOCAL
UNDOTBS1               ONLINE    UNDO            LOCAL
TMPGNX1                ONLINE    TEMPORARY   LOCAL
DATGNX1                ONLINE    PERMANENT   LOCAL
IDXGNX1                ONLINE    PERMANENT   LOCAL
TOOLS                  ONLINE    PERMANENT   LOCAL
DATGNX9700             ONLINE    PERMANENT   LOCAL
IDXGNX9700             ONLINE    PERMANENT   LOCAL
DATGNX9800             ONLINE    PERMANENT   LOCAL
IDXGNX9800             ONLINE    PERMANENT   LOCAL
DATGNX8000             ONLINE    PERMANENT   LOCAL
IDXGNX8000             ONLINE    PERMANENT   LOCAL
```

3. **Pour chacun des schémas Generix**, exporter les données au format DUMP

ATTENTION : IL EST FORTEMENT DECONSEILLE D'EFFECTUER UN EXPORT GLOBAL DE LA BASE INCLUANT TOUS LES SCHEMAS (MODE FULL)

4. Arrêter l'instance Oracle
5. Mettre en commentaire l'instance migrée dans /etc/oratab correspondant à l'ancienne version de Oracle
6. Exporter la base selon les différents schémas
7. Recréer une nouvelle instance Generix identique sur la nouvelle version de Oracle et importer les données. (cf. 2.1)

2.3 Installation de Oracle Text sur une instance existante

1. Si option oracle text a été validée lors de la création de l'instance :

   - Sur tous les schémas existants, lancer le script *ORA_TEXT_PACK.SQL*
   - Effectuer une première alimentation tables Oracle Text :

     Sqlplus <SOCx>/<INFORx
     exec gxp_ot.tie_rec_complete
     exec gxp_ot.pro_rec_complete

   - Mettre en place une stratégie de rafraîchissement complet périodique.

2. Si option oracle text n'a pas été validée lors de la création de l'instance :

   Recréer une nouvelle instance Oracle avec Oracle Text en suivant la même procédure que dans le cas d'une montée de version de Oracle. (cf .2.2)

2.4 Montée de version de Generix

1. Faire la montée de version de Generix (cf.guide d'installation de Generix)
2. Passer « l'alter table » sur la base Oracle
3. Exporter la base selon les différents schémas (dump)
4. Recréer une nouvelle instance Oracle avec Oracle Text en suivant la même procédure que dans le cas d'une montée de version de Oracle. (cf .2.2)
5. Terminer la mise à jour de Generix (GPEV, etc ….)

3 Concepts avancés

3.1 Indexes contextuels

La syntaxe de création d'indexes pour **OT** est inhabituelle. Elle s'opère en 2 phases :

- Création et définition d'un contexte lié à l'index : langue, séparateurs … paramètres de stockage physique. Pour de plus de précisions, se reporter à la documentation Oracle.
- Création de l'index à proprement parlé

Exemple :

```
begin
  ctx_ddl.create_index_set('PRO_REC_INDEX_SET');
  ctx_ddl.add_index('PRO_REC_INDEX_SET','CODSOC, CODPRO');

  ctx_ddl.create_preference('PRO_REC_STORAGE', 'BASIC_STORAGE');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'I_TABLE_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'K_TABLE_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'R_TABLE_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'N_TABLE_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'I_INDEX_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'I_ROWID_INDEX_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');
  ctx_ddl.set_attribute('PRO_REC_STORAGE', 'P_TABLE_CLAUSE',
                        'tablespace XDBGNX1 storage (initial 10M next 2M) nologging');

  ctx_ddl.create_preference('PRO_REC_LEXER', 'BASIC_LEXER');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'printjoins', '/');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'index_themes', 'NO');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'index_text', 'YES');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'base_letter', 'YES');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'index_stems', '4');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'numgroup', '.');
  ctx_ddl.set_attribute('PRO_REC_LEXER', 'theme_language', 'FRENCH');
end;
/

create index PRO_REC_INDEX on PRO_REC (texte) indextype is CTXSYS.CTXCAT
parameters ('index set PRO_REC_INDEX_SET lexer PRO_REC_LEXER storage PRO_REC_STORAGE memory 2M');
```

Exemples de recherches :

➔ recherche d'un mot commençant par « perc »

```
select codsoc, codpro, texte
from pro_rec
where catsearch(texte, 'perc*', 'codsoc=1') > 0
order by codsoc, codpro
```

➔ recherche des mots commençant par « perc » ou « decap »

```
select codsoc, codpro, texte
from pro_rec
where catsearch(texte, 'perc* | decap*', 'codsoc=1') > 0
order by codsoc, codpro
```

➔ recherche des perceuse à percussion et des décapeurs (extrait du jeu standard GENERIX PACK)

```
select codsoc, codpro, texte
from pro_rec
where catsearch(texte, '(perc* percu*) | decap*', 'codsoc=1') > 0
order by codsoc, codpro
```

On utilise systématiquement la fonction **CATSEARCH** (fonction interne Oracle) à laquelle on passe 3 paramètres :

- Le champ de recherche « plan text » : TEXTE
- La valeur à rechercher
- Une partie de clause « where » supplémentaire qui permet de définir sur quelle société on travaille (codsoc=1 dans les exemples ci-dessus)

Cette fonction renvoie une valeur strictement supérieure à **0** lorsqu'un enregistrement est trouvé, on **ne peut pas utiliser un autre opérateur** avec cette fonction.

3.2 Types de recherches potentielles

En plus de la recherche déjà explicitée, il est possible de réaliser des recherches :

- « Avec faute d'orthographe »
- « Phonétique » (algorithme équivalent à la fonction SQL Soundex)

La plus intéressante est évidemment la recherche phonétique, en particulier pour des recherches de tiers dans la base de données. Par exemple avec ce type de recherche, les mots « DUPONT » , « DUPOND » et « DUPON » sont strictement équivalents.

4 Mise en place du rafraichissement

Il est possible d'utiliser les procédures périodiques GENERIX (UPRC) pour réaliser les rafraichissements, ainsi que les jobs Oracle. Suivant le contexte client, ceci doit être paramétré sur site (modalité et fréquence).

4.1 Premier rafraichissement

Afin de valider le bon fonctionnement, il est nécessaire d'exécuter une première fois l'alimentation intégrale des tables Oracle Text. Pour cela, il faut exécuter le script suivant sous SQL*Plus :

```
exec gxp_ot.tie_rec_complete ;
exec gxp_ot.pro_rec_complete ;
```

Dans le cas ou l'on ne souhaite pas faire de recherches OT sur les tiers par exemple, seule la 2ième ligne est exécuter bien entendu.

4.2 Exemple d'un rafraichissement complet périodique

L'exploitation des jobs Oracle impose que les paramètres suivant soient configurés correctement en fonction du contexte dans init*xxx*.ora (ou spfile si utilisé) :

- Job_queue_processes :
- Job_queue_interval :

Exemple pour augmenter le nombre de jobs autorisés si on gère le spfile (paramètre dynamique, dans le cas d'une gestion par init*xxx*.ora il faut redémarrer la base) :

```
alter system set job_queue_processes = 10 scope=both ;
```

Hypothèses :

- Rafraichissement complet de la table TIE_REC
- Période de 24 heures
- Les commandes sont exécutées sous SQL*Plus

Script de création du job :

```
set serveroutput on size 10000
declare
  exec varchar2(256);
  -- Définition de l'unité de temps (1 seconde)
  tpsu number := (1/(24*60*60));
  -- Calcul de la période
  tps  number := tpsu * 60*60*24;
  -- Retour
  x    integer;
begin
  dbms_output.enable(10000);
```

```
  dbms_output.put_line('Intervalle en secondes : '||round(tps*60*60*24));
  exec := 'gxp_ot.tie_rec_complete;';
  dbms_job.submit(job => x,
                  what => exec,
                  next_date => SYSDATE,
                  interval => 'SYSDATE + '||tps);
  dbms_output.put_line('Numéro de job: '||x);
end;
/

commit;
```

Remarque : la création d'un job doit être validée (commit) à l'extérieur du bloc PL/SQL.

Liste des jobs :

```
select job Numero,
      substr(schema_user,1,10) Schema,
      substr(what,1,20) Commande,
      total_time tps_exec_sec,
      nvl(to_char(last_date,'dd/mm/yyyy hh24:mi:ss'),'Jamais execute') derniere_exec,
      to_char(next_date,'dd/mm/yyyy hh24:mi:ss') prochaine_exec,
      decode(broken,'Y','Job Arrêté','Job Valide') Valide
from user_jobs;
```

Suppression d'un job :

```
exec dbms_job.remove(numero de job) ;

commit ;
```

Remarque : le numéro correspond au numéro de job de la liste ci-dessus.

4.3 Exemple d'un rafraichissement différentiel périodique

Dans ce cadre, il faut activer les triggers livrés dans le script afin d'alimenter les tables intermédiaires (TIE_REC_MOD et PRO_REC_MOD) :

```
alter trigger tw_tie_rec_mod enable ;
alter trigger tw_pro_rec_mod enable ;
```

Hypothèses :

- Rafraichissement différentiel de la table TIE_REC
- Période de 10 minutes
- Les commandes ci-dessous sont exécutées sous SQL*Plus

Script de création du job :

```
set serveroutput on size 10000
declare
  exec varchar2(256);
  -- Définition de l'unité de temps (1 seconde)
  tpsu number := (1/(24*60*60));
  -- Calcul de la période
  tps  number := tpsu * 60*10;
  -- Retour
  x    integer;
begin
  dbms_output.enable(10000);
  dbms_output.put_line('Intervalle en secondes : '||round(tps*60*60*24));
  exec := 'gxp_ot.tie_rec_fast;';
  dbms_job.submit(job => x,
                what => exec,
                next_date => SYSDATE,
                interval => 'SYSDATE + '||tps);
  dbms_output.put_line('Numéro de job: '||x);
end;
/

commit;
```

Remarque : la création d'un job doit être validée (commit) à l'extérieur du bloc PL/SQL.

La liste et la suppression des jobs se font comme décrit dans la section précédente.