Module Administration BD : Paquetages PL/SQL, SQL dynamique et méta-schéma

1. Paquetages

La surcouche procédurale PL/SQL permet également l'utilisation de paquetages prédéfinis, ainsi que la définition de nouveaux paquetages. Nous précisons ci-dessous les intérêts d'une telle construction :

— sécurité et contrôle au sein des bases de données
— performance, efficacité des applicatifs mis en place
— modularité, clarté, flexibilité et évolutivité des applicatifs

Nous proposons un exemple de définition de paquetage, portant sur un cas d'école, avec des méthodes de conversion de monnaie à monnaie.
La spécification d'un paquetage comprend deux parties distinctes. Dans un premier temps, le paquetage est déclaré, les variables et les méthodes (fonctions comme procédures) sont publiées et rendues ainsi publiques. Dans un second temps, le paquetage est spécifié, les variables privées, les méthodes privées ainsi que le corps des méthodes publiques sont alors définis.

1.1 Illustration

La déclaration du paquetage **Finances** est proposée comme suit :

```
create or replace package Finances
 as
vTx_EF constant number := 6.55957;
vTx_ED constant number := 1.3926;

 function conversionF_EF (euros in number) return number;
 procedure conversionP_EF (euros in number, francs out number);
 function conversionF_ED (euros in number) return number;
end Finances ;
 /
```

Listing 1 – Déclaration paquetage

La spécification du paquetage **Finances** fait l'objet d'un autre script, une fonction privée *conversion* est donnée à vertu d'exemple.

```
create or replace package body Finances
as

function conversion (montant in number, taux in number)
return number
```

```
is
begin
return (round(montant * taux, 2));
Exception when OTHERS then return null;
end;

function conversionF_ED (euros in number)
return number is
begin
return conversion (euros, vTx_ED);
end;

function conversionF_EF (euros in number)
return number is
begin
return conversion (euros, vTx_EF);
end;


procedure conversionP_EF (euros in number, francs out number)
is
begin
francs := (round(euros * vTx_EF, 2));
Exception when OTHERS then dbms_output.put_line(' erreur argument ');
end;

end Finances;
/
```

Listing 2 – Corps du paquetage

1.2 Utilisation

Des exemple d'utilisation sont fournis :

```
select salaire, Finances.conversionF_ED(salaire) as enDollars,
Finances.conversionF_EF(salaire) as enFrancs from employe;

select * from employe where Finances.conversionF_EF(salaire) > 10000 ;
```

Listing 3 – Quelques usages possibles

1.3 Paquetages Oracle prédéfinis

Des exemples de paquetages fournis par Oracle (*built_in packages*) sont listés, de manière largement non exhaustive, ici :

- DBMS_OUTPUT : stocke l'information dans une mémoire tampon (buffer) afin de la restituer ultérieurement
- DBMS_METADATA : retourne l'ensemble des informations (définitions des objets du schéma) provenant du dictionnaire de données
- UTL_FILE : fournit les éléments nécessaires à des opérations de lecture/écriture dans des fichiers textes externes à la BD
- DBMS_ALERT : notifier des évènements aux usagers de la BD
- DBMS_SQL : fonctionnalités SQL étendues
- DBMS_XMLGEN : proposer les tables au format XML

— DBMS_STATS : collecter des statistiques sur les schémas (très utile pour l'optimisation des accès)
— DBMS_UTILITY : fonctions utilitaires : temps, conversion, analyse en routine ...

Le contenu de chaque paquetage est consultable au travers de la commande DESCRIBE.

```
desc dbms_metadata
```

Listing 4 – Description paquetage

1.4 Les exceptions prédéfinies

PL/SQL gère une collection d'erreurs prédéfinies qui peuvent être traitées dans la section EXCEPTION et donner lieu à divers comportements spécifiques. Ces erreurs possèdent un code erreur (SQLCODE) et un message d'erreur (SQLERRM) dédié.

```
DECLARE
empTup EMP%ROWTYPE;
numEmp EMP.num%type;
salaireNul EMP.salaire%type;
salExc exception;
BEGIN
FOR numEmp IN 1..21
LOOP
SELECT * INTO empTup FROM EMP WHERE num = numEmp;
INSERT INTO EMP (num, nom, fonction) VALUES (empTup.num, empTup.nom, empTup.fonction);
if (empTup.salaire is null)
then raise salExc;
end if;
END LOOP;
EXCEPTION
WHEN salExc THEN DBMS_OUTPUT.PUT_LINE( 'salaire absent');
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE( 'num inexistant ' || numEmp || ' '||
    SQLERRM);
WHEN TOO_MANY_ROWS THEN DBMS_OUTPUT.PUT_LINE( 'attention au schema !');
WHEN DUP_VAL_ON_INDEX THEN DBMS_OUTPUT.PUT_LINE( 'Contrainte PK Message SQL : ' ||
    SQLERRM);
WHEN OTHERS THEN DBMS_OUTPUT.PUT_LINE( 'Contrainte PK Message SQL : ' || SQLCODE);
END;
/
```

Listing 5 – Illustration exceptions

2. SQL dynamique

Le SQL dit "dynamique" est un passage obligé en PL/SQL dans deux cas de figures précis :

1. répétitivité d'un traitement avec prise en charge de valeurs différentes à l'exécution (définition d'un code générique et réutilisable)

```
variable reg varchar2(10);
execute :reg := '91';
SELECT chef_lieu FROM region WHERE reg = :reg;
```

Listing 6 – Attachement de variable

2. recours aux instructions DDL : `CREATE, DROP ou ALTER TABLE` en particulier

L'ordre le plus exploité en est `EXECUTE IMMEDIATE` et porte sur une clause SQL (entre simples quotes).

```
create table test (
 a varchar(5), b integer);

create or replace procedure supp (tble_name in varchar)
 is
 begin
 execute immediate 'drop table '||tble_name||' cascade
 constraints';
 dbms_output.put_line('table '||tble_name||' detruite ');
 exception
 when others then dbms_output.put_line('Pb sur la suppression ');
 end;
 /

execute supp('test');
```

Listing 7 – Suppression de table

Les variables attachées offrent aussi un moyen de lutter contre les tentatives de SQL injection en permettant de passer les valeurs d'une condition de selection de manière indirecte.

3. Dictionnaire de données ou méta-schéma

Un ensemble de vues (avec pour propriétaire l'utilisateur SYS) permettent de disposer d'une vue d'ensemble sur, à la fois, les schémas utilisateurs (vues statiques ou structurelles) et les activités courantes menées au sein de la base de données (vues dynamiques).

4. Vues statiques

Les éléments du schéma sont par exemple envisagés comme des métadonnées et consultables comme des données à part entière. Différents types de vues sont exploitables et répondent à trois grands niveaux organisationnels :

1. USER_ : Informations sur tous les objets du schéma utilisateur
2. ALL_ : Informations sur tous les objets qui sont accessibles à l'utilisateur
3. DBA_ : Informations sur tous les objets de la base

Une vue des vues permet de lister toutes les vues du dictionnaire : DICT (ou DICTIONARY).

```
select table_name from user_tables;

select constraint_name, table_name, column_name from user_cons_columns;
```

Listing 8 – Exemples vues statiques

5. Vues dynamiques

Elles fournissent divers renseignements sur l'état de la base de données. Il est ainsi possible d'en savoir plus sur les différentes ressources allouées ou sur les activités courantes des usagers. Ces vues ont pour préfixe V_$ ou bien V$.

```
select username, terminal, program from v$process;
desc v$lock
```

Listing 9 – Exemples vues statiques