Base de données réparties

Transparence d'emplacement

Après avoir crée les liens de bases de données, plusieurs objets :

- Les vues.
- Les synonymes
- Les procédures stockées

peuvent servir à cacher la distribution des données aux utilisateurs :

Transparence d'emplacement

➢ Les vues.

Les vues peuvent fournir une transparence par rapport aux tables locales et distantes.

Par exemple :

- La table *Employé est sur une BD locale*
- *La table Département est sur une BD distante.*

➢ **Rendre les tables transparentes**:

Créer une vue dans la BD locale qui fait la jointure des données locales et distantes,

Transparence d'emplacement

- Exemple:

Create view client_casa as select * from client@lien_casa;

select * from client_casa;

Synonymes

Les synonymes sont des noms simples qui permettent d'identifier de façon unique dans un système distribué les objets qu'ils nomment.

Les synonymes peuvent être crées pour différents objets :

- Tables
- Types
- Vues
- Procédures
- Fonctions
- Packages.

Synonymes

Syntaxe :

CREATE [PUBLIC] SYNONYM nom_synonyme For objet[@nom-lien-BD]

- L'utilisateur a besoin de connaître uniquement le nom du synonyme vers lequel il pointe.
- Un synonyme public est partagé par tous les utilisateurs alors qu'un synonyme privé est reconnu que par un compte individuel.

Synonymes

Exemple :

Créer le synonyme de la table **Employe_casa de la base de données DBCasa spécifiée par le lien « lien_casa » sur la** base de données locale de **Fes :**

CREATE PUBLIC SYNONYM Cl_casa For Employe_casa@lien_casa

- Quel est le nombre de clients de Casa? – Requête exécutée sur Fes!!!!!

Select count(*) From Cl_Casa

Synonymes

Suppression de synonymes :

DROP [PUBLIC] SYNONYM nom_synonyme

Tables :

User_synonyms, All_synonyms ,

Exercice

Ecrire en SQL la requête suivante « On recherche les employés affectés au département génie logiciel » dans les deux cas suivants :

- La requête est émise du site A où se trouve la table *Employé*.
- La requête est émise du site B où se trouve la table *Département*.

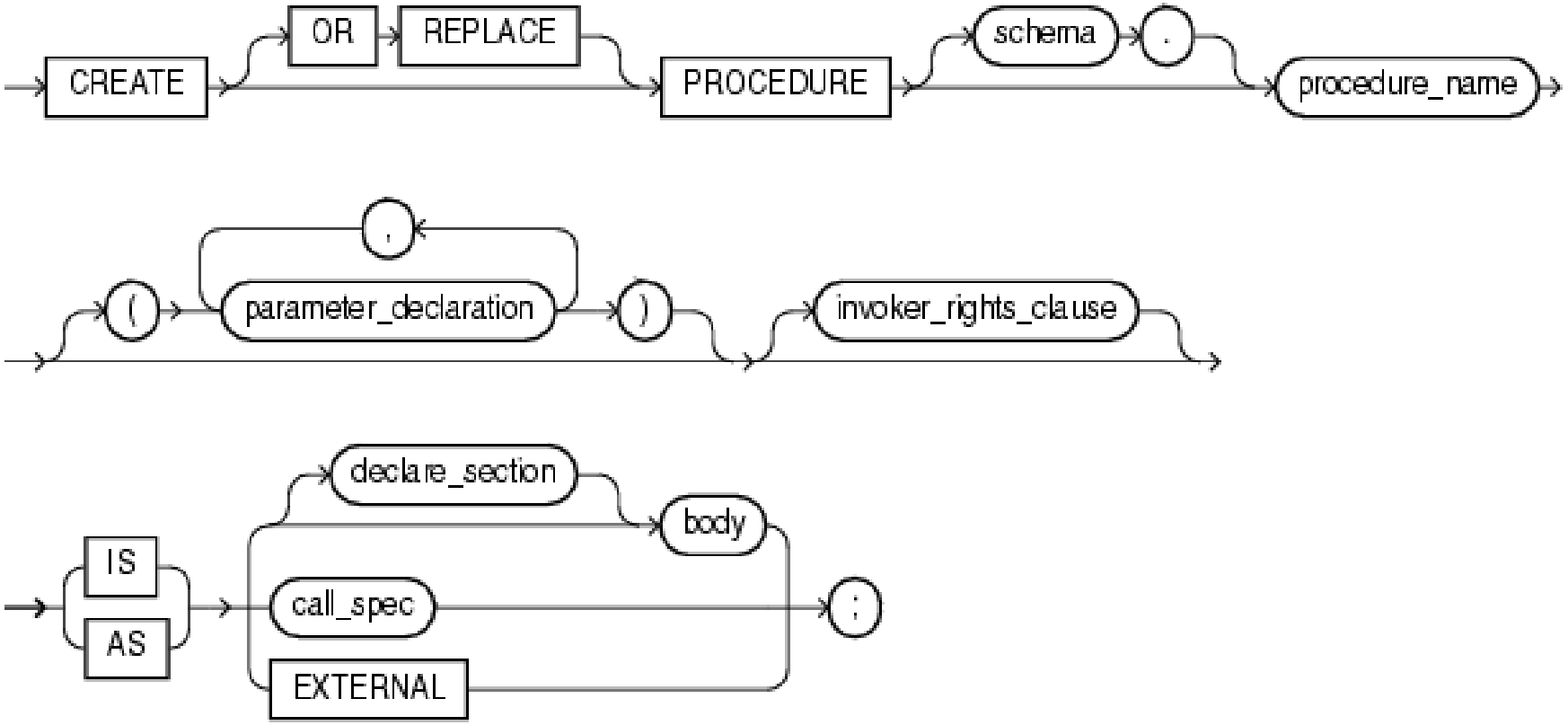
Procédures

Les unités de programmes PL/SQL, peuvent servir à:

1. *Référer à des données distantes,*
2. *Appeler des procédures distantes*
3. *Utiliser des synonymes pour référer à des procédures distantes.*

Procédures

Syntaxe :



Procédures

Référer à des données distantes

Considérons la procédure *LicencierEmployé :*

```
CREATE PROCEDURE LicencierEmployé (enum NUMBER)
AS
BEGIN
DELETE FROM Employé@hq.acme.com
WHERE NumEmp = enum
END;
```

Quand l'utilisateur ou l'application appelle la procédure *LicencierEmployé,* il/elle ne voit pas que la table distante est en cours de mise à jour.

Procédures

Utiliser des synonymes pour référer à des procédures distantes

Nous pouvons également créer un synonyme pour le lien à la table *Employé*.

```
CREATE SYNONYM Emp FOR Employe@hq.acme.com;
```

La procédure *LicencierEmployé devient :*

```
CREATE PROCEDURE LicencierEmployé (enum NUMBER)
AS
BEGIN
DELETE FROM Emp WHERE NumEmp = enum;
END;
```

Réplication des données

- Commande COPY

La première option consiste à répliquer régulièrement les données sur le serveur local au moyen de la commande COPY de SQL*Plus.

COPY {FROM BD_source | TO BD_destination | FROM BD_source TO BD_destination}
{CREATE|INSERT| APPEND| REPLACE}
table_destination[(colonne1, colonne2, ...)]
USING requête_fragmentation

Réplication des données

- **CREATE : Si la table destination existe, copy génère une erreur. Sinon, la table est créée et les** données seront copiées : **CREATE + INSERT**
- **INSERT : Si la table destination existe, copy ajoute les nouvelles lignes. Sinon, elle génère une erreur** et s'arrête : **INSERT**
- **APPEND : Si la table destination existe, copy ajoute les données. Sinon, elle crée la table et insert les** données. : **[CREATE] + INSERT**
- **REPLACE : Si la table destination existe, copy supprime et recrée la table avec les nouvelles données.** Sinon, elle crée la table et insert les données : **[DROP] + CREATE + INSERT**
- L'inconvénient est que les données ne peuvent pas être mises à jour

Copie de données distantes sous oracle

Exemples :

Duplication

COPY FROM user2/user2@Rabat TO user2/user2@Casa CREATE ***client USING (Select N_client,***nom_client, Ville FROM client **)**

Fragmentation horizontale

COPY FROM user2/user2@Rabat TO user2/user2@Casa CREATE ***Client_casa USING (Select N_compte,*** Client.N_agence, Nom, Adresse, Solde **From Client, Agence Where Client.N_agence = Agence.N_agence and** Ville='Casa' **)**

Copie de données distantes sous oracle

Exemples :

Fragmentation verticale

COPY FROM said/said@Rabat TO said/said@Casa APPEND *Client_info USING (Select N_compte,* solde From Client)

Fragmentation verticale

COPY FROM said/said@Rabat TO said/said@Casa REPLACE *Client_info USING (Select N_compte,* solde From Client)