12/07/2023

SUPPORT PRATIQUE PL / SQL

Joachim ZADI

Partie 1

L'objectif de ces exercices est de créer des tables, leur clé primaire et des contraintes de vérification (`NOT NULL` et `CHECK`). La première partie des exercices (de 1.1 à 1.4 concerne la base Parc Informatique). Le dernier exercice traite d'une autre base (Chantiers) qui s'appliquera à une base 11g.

Exercice 11 : Présentation de la base de données

Une entreprise désire gérer son parc informatique à l'aide d'une base de données. Le bâtiment est composé de trois étages. Chaque étage possède son réseau (ou segment distinct) Ethernet. Ces réseaux traversent des salles équipées de postes de travail. Un poste de travail est une machine sur laquelle sont installés certains logiciels. Quatre catégories de postes de travail sont recensées (stations Unix, terminaux X, PC Windows et PC NT). La base de données devra aussi décrire les installations de logiciels.

Les noms et types des colonnes sont les suivants :

Colonne	Commentaires	Types
indIP	trois premiers groupes IP (exemple : 130.120.80)	VARCHAR2(11)
nomSegment	nom du segment	VARCHAR2(20)
etage	étage du segment	NUMBER(2)
nSalle	numéro de la salle	VARCHAR2(7)
nomSalle	nom de la salle	VARCHAR2(20)
nbPoste	nombre de postes de travail dans la salle	NUMBER(2)
nPoste	code du poste de travail	VARCHAR2(7)
nomPoste	nom du poste de travail	VARCHAR2(20)
ad	dernier groupe de chiffres IP (exemple : 11)	VARCHAR2(3)
typePoste	type du poste (Unix, TX, PCWS, PCNT)	VARCHAR2(9)
dateIns	date d'installation du logiciel sur le poste	DATE
nLog	code du logiciel	VARCHAR2(5)
nomLog	nom du logiciel	VARCHAR2(20)
dateAch	date d'achat du logiciel	DATE
version	version du logiciel	VARCHAR2(7)
typeLog	type du logiciel (Unix, TX, PCWS, PCNT)	VARCHAR2(9)
prix	prix du logiciel	NUMBER(6,2)
numIns	numéro séquentiel des installations	NUMBER(5)
dateIns	date d'installation du logiciel	DATE
delai	intervalle entre achat et installation	INTERVAL DAY(5) TO SECOND(2),
typeLP	types des logiciels et des postes	VARCHAR2(9)
nomType	noms des types (Terminaux X, PC Windows…)	VARCHAR2(20)

Figure 1: Caractéristiques des colonnes

Exercice 12 : Création des tables

Écrivez puis exécutez le script SQL (que vous appellerez creParc.sql) de création des tables avec leur clé primaire (en gras dans le schéma suivant) et les contraintes suivantes :

- Les noms des segments, des salles et des postes sont non nuls.
- Le domaine de valeurs de la colonne ad s'étend de 0 à 255.
- La colonne prix est supérieure ou égale à 0.
- La colonne dateIns est égale à la date du jour par défaut.

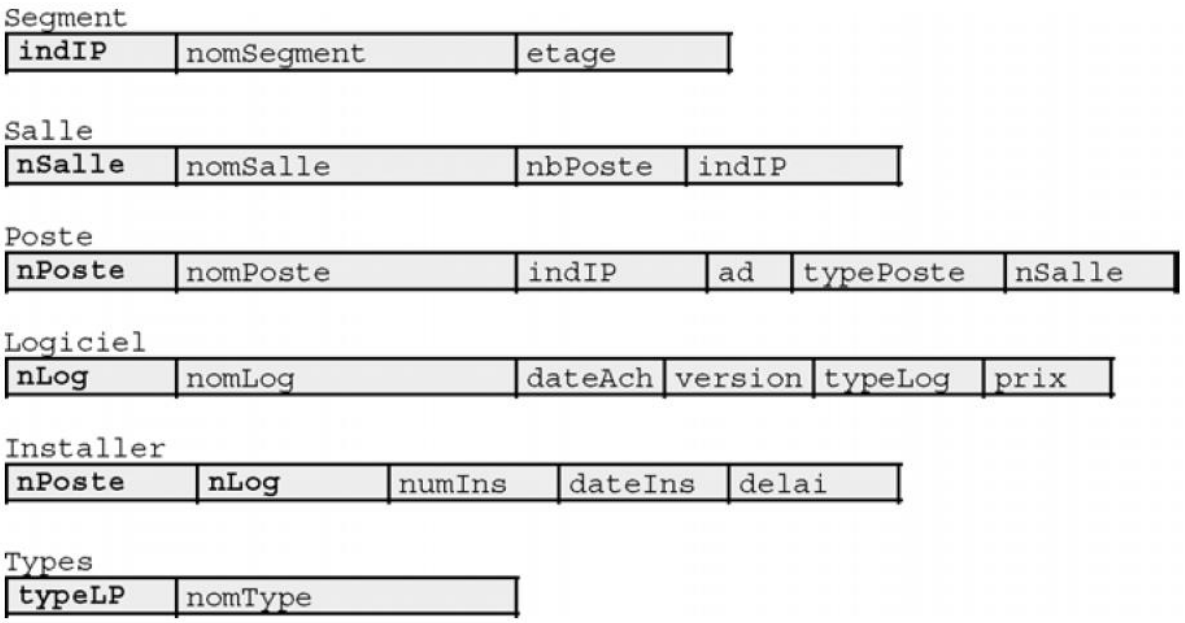

Figure 2: Schéma des tables

Exercice 13 : Structure des tables

Écrivez puis exécutez le script SQL (que vous appellerez descParc.sql) qui affiche la description de toutes ces tables (en utilisant des commandes DESC). Comparer avec le schéma.

Exercice 14 : destruction des tables

Écrivez puis exécutez le script SQL de destruction des tables (que vous appellerez dropParc.sql). Lancer ce script puis à nouveau celui de la création des tables.

Exercice 15 : schéma de la base chantier

Une société désire informatiser les visites des chantiers de ses employés. Pour définir cette base de données, une première étude fait apparaître les informations suivantes :

- Chaque employé est modélisé par un numéro, un nom et une qualification.
- Un chantier est caractérisé par un numéro, un nom et une adresse.
- L'entreprise dispose de véhicules pour lesquels est important de stocker pour le numéro d'immatriculation, le type (un code valant par exemple 0 pour une camionnette, 1 pour une moto et 2 pour une voiture) ainsi que le kilométrage en fin d'année.
- Le gestionnaire a besoin de connaître les distances parcourues par un véhicule pour chaque visite d'un chantier.
- Chaque jour, un seul employé sera désigné conducteur des visites d'un véhicule.
- Pour chaque visite, il est important de pouvoir connaître les employés transportés.

Les colonnes à utiliser sont les suivantes :

Colonne	Commentaires	Types
kilometres	kilométrage d'un véhicule lors d'une sortie	NUMBER
n_conducteur	numéro de l'empoyé conducteur	VARCHAR2(4)
n_transporte	numéro de l'empoyé transporté	VARCHAR2(4)

Figure 3: Caractéristiques des colonnes à ajouter

L'exercice consiste à compléter le schéma relationnel ci-après (ajout de colonnes et définition des contraintes de clé primaire et étrangère).

```
CREATE TABLE employe (n_emp VARCHAR(4),nom_emp VARCHAR(20),
 qualif_emp VARCHAR(12), CONSTRAINT pk_emp PRIMARY KEY(n_emp));

CREATE TABLE chantier (n_chantier VARCHAR(10), nom_ch VARCHAR(10),
  adresse_ch VARCHAR(15), CONSTRAINT pk_chan PRIMARY KEY(n_chantier));

CREATE TABLE vehicule (n_vehicule VARCHAR(10), type_vehicule VARCHAR(1),
  kilometrage NUMBER, CONSTRAINT pk_vehi PRIMARY KEY(n_vehicule));

CREATE TABLE visite(n_chantier VARCHAR(10), n_vehicule VARCHAR(10),
 date_jour DATE,  ...
 CONSTRAINT pk_visite PRIMARY KEY(...),
 CONSTRAINT fk_depl_chantier FOREIGN KEY(n_chantier) ...,
 CONSTRAINT fk_depl_vehicule FOREIGN KEY(n_vehicule)  ...,
 CONSTRAINT fk_depl_employe FOREIGN KEY(n_conducteur) ... );

CREATE TABLE transporter (...
 CONSTRAINT pk_transporter PRIMARY KEY (...),
 CONSTRAINT fk_transp_visite FOREIGN KEY ... ,
 CONSTRAINT fk_transp_employe FOREIGN KEY ...);
```

Partie 2

Les objectifs des premiers exercices sont :

- *d'insérer des données dans les tables du schéma Parc Informatique et du schéma des chantiers*
- *de créer une séquence et d'insérer des données en utilisant une séquence ;*
- *de modifier des données.*

Exercice 2.1 : Insertion de données

Écrivez puis exécutez le script SQL (que vous appellerez insParc.sql) afin d'insérer les données dans les tables suivantes :

Table : Segment

INDIP	NOMSEGMENT	ETAGE
130.120.80	Brin RDC	
130.120.81	Brin 1er étage	
130.120.82	Brin 2ème étage	

Table : Salle

NSALLE	NOMSALLE	NBPOSTE	INDIP
s01	Salle 1	3	130.120.80
s02	Salle 2	2	130.120.80
s03	Salle 3	2	130.120.80
s11	Salle 11	2	130.120.81
s12	Salle 12	1	130.120.81
s21	Salle 21	2	130.120.82
s22	Salle 22	0	130.120.83
s23	Salle 23	0	130.120.83

Table : Poste

NPOSTE	NOMPOSTE	INDIP	AD	TYPEPOSTE	NSALLE
p1	Poste 1	130.120.80	01	TX	s01
p2	Poste 2	130.120.80	02	UNIX	s01
p3	Poste 3	130.120.80	03	TX	s01
p4	Poste 4	130.120.80	04	PCWS	s02
p5	Poste 5	130.120.80	05	PCWS	s02
p6	Poste 6	130.120.80	06	UNIX	s03
p7	Poste 7	130.120.80	07	TX	s03
p8	Poste 8	130.120.81	01	UNIX	s11
p9	Poste 9	130.120.81	02	TX	s11
p10	Poste 10	130.120.81	03	UNIX	s12
p11	Poste 11	130.120.82	01	PCNT	s21
p12	Poste 12	130.120.82	02	PCWS	s21

Figure 4: Données des tables

Table : Logiciel

NLOG	NOMLOG	DATEACH	VERSION	TYPELOG	PRIX
log1	Oracle 6	13/05/95	6.2	UNIX	3000
log2	Oracle 8	15/09/99	8i	UNIX	5600
log3	SQL Server	12/04/98	7	PCNT	2700
log4	Front Page	03/06/97	5	PCWS	500
log5	WinDev	12/05/97	5	PCWS	750
log6	SQL*Net		2.0	UNIX	500
log7	I. I. S.	12/04/02	2	PCNT	810
log8	DreamWeaver	21/09/03	2.0	BeOS	1400

Table : Types

TYPELP	NOMTYPE
TX	Terminal X-Window
UNIX	Système Unix
PCNT	PC Windows NT
PCWS	PC Windows
NC	Network Computer

Figure 5 : Données des tables suites

Exercice 22 : Gestion d'une séquence

Dans ce même script, créez la séquence `sequenceIns` commençant à la valeur 1, d'incrément 1, de valeur maximale 10 000 et sans cycle. Utilisez cette séquence pour estimer la colonne `numIns` de la table `Installer`. Insérez les enregistrements suivants :

Table : Installer

NPOSTE	NLOG	NUMINS	DATEINS	DELAI
p2	log1	1	15/05/03	
p2	log2	2	17/09/03	
p4	log5	3		
p6	log6	4	20/05/03	
p6	log1	5	20/05/03	
p8	log2	6	19/05/03	
p8	log6	7	20/05/03	
p11	log3	8	20/04/03	
p12	log4	9	20/04/03	
p11	log7	10	20/04/03	
p7	log7	11	01/04/02	

Figure 6: Données de la table Installer

Exercice 23 : Modification de données

Écrivez le script `modification.sql`, qui permet de modifier (avec `UPDATE`) la colonne `etage` (pour l'instant nulle) de la table `Segment` afin d'affecter un numéro d'étage correct (0 pour le segment 130.120.80, 1 pour le segment 130.120.81, 2 pour le segment 130.120.82).

Diminuez de 10 % le prix des logiciels de type 'PCNT'.

Vérifiez :

```
SELECT * FROM Segment;
SELECT nLog, typeLog, prix FROM Logiciel;
```

Exercice 24 : insertion dans la base Chantier

Écrivez puis exécutez le script SQL (que vous appellerez `insChantier.sql`) afin d'insérer les données suivantes :

- une dizaine d'employés (numéros E1 à E10) en considérant diverses qualifications (OS, Assistant, Ingénieur et Architecte) ;
- quatre chantiers et cinq véhicules ;
- deux ou trois visites de différents chantiers durant trois jours ;
- la composition (de un à trois employés transportés) de chaque visite.

Partie 3

Les objectifs de ces exercices sont :

- *d'ajouter et de modifier des colonnes ;*
- *d'ajouter des contraintes ;*
- *de traiter les rejets.*

Exercice 31 : Ajout de colonnes

Écrivez le script `évolution.sql` qui contient les instructions nécessaires pour ajouter les colonnes suivantes (avec `ALTER TABLE`). Le contenu de ces colonnes sera modifié ultérieurement.

Table	Nom, type et signification des nouvelles colonnes
`Segment`	`nbSalle NUMBER(2)` : nombre de salles `nbPoste NUMBER(2)` : nombre de postes
`Logiciel`	`nbInstall NUMBER(2)` : nombre d'installations
`Poste`	`nbLog NUMBER(2)` : nombre de logiciels installés

Figure 7 : Données de la table Installer

Vérifier la structure et le contenu de chaque table avec DESC et SELECT.

Exercice 32 : Modification de colonnes

Dans ce même script, ajoutez les instructions nécessaires pour :

- augmenter la taille dans la table `Salle` de la colonne `nomSalle` (passer à `VARCHAR2(30)`) ;
- diminuer la taille dans la table `Segment` de la colonne `nomSegment` à `VARCHAR2(15)` ;
- tenter de diminuer la taille dans la table `Segment` de la colonne `nomSegment` à `VARCHAR2(14)`.

Pourquoi la commande n'est-elle pas possible ?

Vérifiez par `DESC` la nouvelle structure des deux tables.

Vérifiez le contenu des tables :

```
SELECT * FROM Salle;
SELECT * FROM Segment;
```

Exercice 33 : Ajout de contraintes

Ajoutez les contraintes de clés étrangères pour assurer l'intégrité référentielle entre les tables suivantes (avec `ALTER TABLE… ADD CONSTRAINT…`). Adoptez les conventions recommandées dans le chapitre 1 (comme indiqué pour la contrainte entre `Poste` et `Types`).

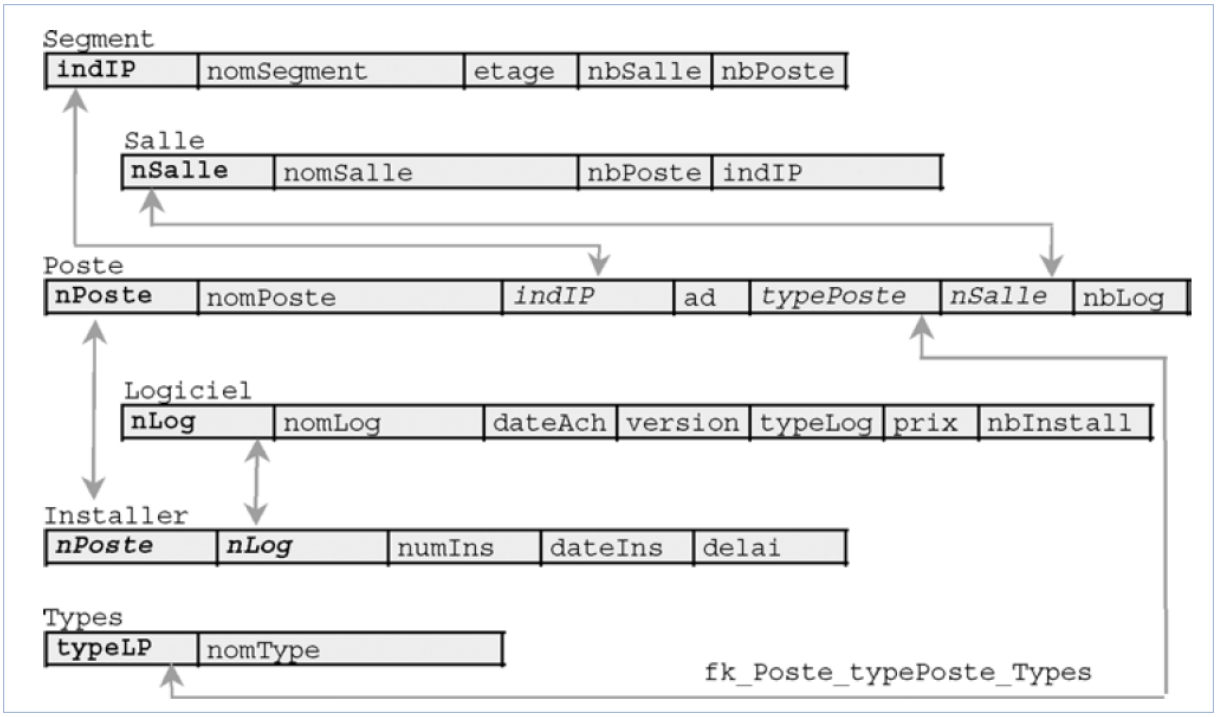

Figure 8 : Contraintes référentielles à créer

Si l'ajout d'une contrainte référentielle renvoie une erreur, vérifier les enregistrements des tables « pères » et « fils » (notamment au niveau de la casse des chaînes de caractères, 'Tx' est différent de 'TX' par exemple).

Modifiez le script SQL de destruction des tables (`dropParc.sql`) en fonction des nouvelles contraintes.

Lancer ce script puis tous ceux écrits jusqu'ici.

Exercice 34 : Traitements des rejets

Créez la table `Rejets` avec la structure suivante (ne pas mettre de clé primaire) :

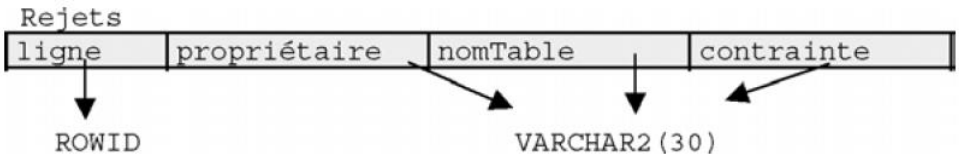

Figure 9 : Table des rejets (exceptions)

Cette table permettra de retrouver les enregistrements qui ne vérifient pas de contraintes lors de la réactivation.

Ajoutez les contraintes de clés étrangères entre les tables `Salle` et `Segment` et entre `Logiciel` et `Types` (en gras dans le schéma suivant). Utilisez la directive `EXCEPTIONS INTO` pour récupérer des informations sur les erreurs.

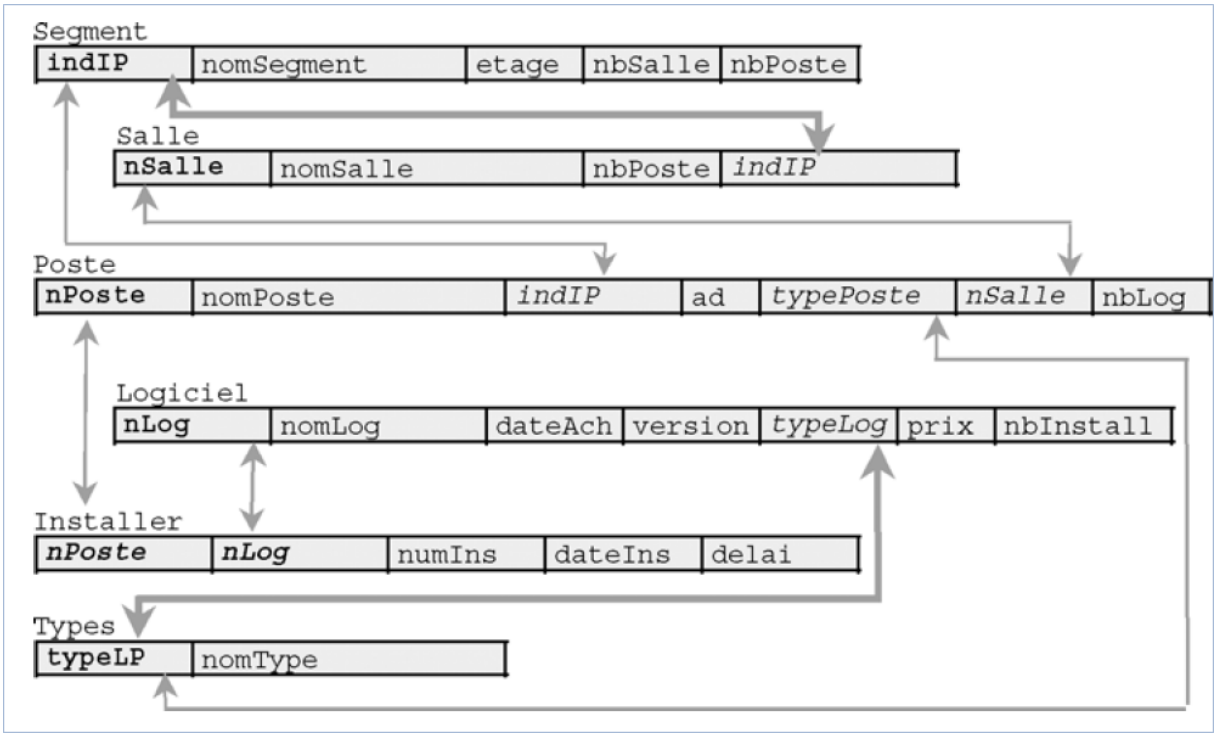

Figure 10 : Contraintes référentielles à créer

La création de ces contraintes doit renvoyer une erreur car :

- il existe des salles ('s22' et 's23') ayant un numéro de segment qui n'est pas référencé dans la table `Segment` ;
- il existe un logiciel ('log8') dont le type n'est pas référencé dans la table `Types`.

Vérifiez dans la table `Rejets` les enregistrements qui posent problème. Vérifier la correspondance avec les `ROWID` des tables `Salle` et `Logiciel` :

```
SELECT * FROM Rejets;
SELECT ROWID,s.* FROM Salle s
       WHERE ROWID IN (SELECT ligne FROM Rejets);
SELECT ROWID,l.* FROM Logiciel l
       WHERE ROWID IN (SELECT ligne FROM Rejets);
```

Supprimez les enregistrements de la table `Rejets`.

Supprimez les enregistrements de la table `Salle` qui posent problème. Ajouter le type de logiciel ('`BeOS`', '`Système Be`') dans la table `Types`.

Exécutez à nouveau l'ajout des deux contraintes de clé étrangère. Vérifier que les instructions ne renvoient plus d'erreur et que la table `Rejets` reste vide.

Exercice 35 : Ajout de colonnes dans la base *Chantiers*

Écrivez le script `évolChantier.sql` qui modifie la base *Chantiers* afin de pouvoir stocker :

- la capacité en nombre de places de chaque véhicule ;
- la liste des types de véhicule interdits de visite concernant certains chantiers ;
- la liste des employés autorisés à conduire certains types de véhicule ;
- le temps de trajet pour chaque visite (basé sur une vitesse moyenne de 40 kilomètres par heure).

Vous utiliserez une colonne virtuelle.

Exercice 36 : Mise à jour de la base *Chantiers*

Écrivez le script `majChantier.sql` qui met à jour les nouvelles colonnes de la base *Chantiers* de la manière suivante :

- affectation automatique du nombre de places disponibles pour chaque véhicule (1 pour les motos, 3 pour les voitures et 6 pour les camionnettes) ;
- déclaration d'un chantier inaccessible pour une camionnette et d'un autre inaccessible aux motos ;
- déclaration de diverses autorisations pour chaque conducteur (affecter toutes les autorisations à un seul conducteur).

Vérifiez le contenu de chaque table (et de la colonne virtuelle) avec `SELECT`.

Partie 4

Les objectifs de ces exercices sont :

- *de créer dynamiquement des tables et leurs données ;*
- *d'écrire des requêtes monotables et multitables ;*
- *de réaliser des modifications synchronisées ;*
- *de composer des jointures et des divisions.*

Exercice 41 : Création dynamique de tables

Écrivez le script `créaDynamique.sql` permettant de créer les tables `Softs` et `PCSeuls` suivantes (en utilisant la directive `AS SELECT` de la commande `CREATE TABLE`). Vous ne poserez aucune contrainte sur ces tables.

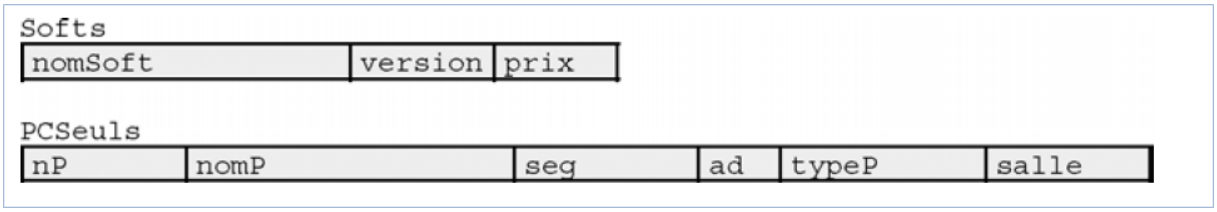

Figure 11 : Structures des nouvelles tables

La table `Softs` sera construite sur la base de tous les enregistrements de la table `Logiciel` que vous avez créée et alimentée précédemment.

La table `PCSeuls` doit seulement contenir les enregistrements de la table `Poste` qui sont de type 'PCWS' ou 'PCNT'.

Vérifier :

```
SELECT * FROM Softs;
SELECT * FROM PCSeuls;
```

Exercice 42 : Requêtes monotables

Écrivez le script `requêtes.sql`, permettant d'extraire, à l'aide d'instructions `SELECT`, les données suivantes :

1. Type du poste 'p8'.
2. Noms des logiciels Unix.
3. Nom, adresse IP, numéro de salle des postes de type 'Unix' ou 'PCWS'.
4. Même requête pour les postes du segment '130.120.80' triés par numéros de salles décroissants.
5. Numéros des logiciels installés sur le poste 'p6'.
6. Numéros des postes qui hébergent le logiciel 'log1'.
7. Nom et adresse IP complète (ex : '130.120.80.01') des postes de type TX (utiliser l'opérateur de concaténation).

Exercice 43 : Fonctions et groupements

8. Pour chaque poste, le nombre de logiciels installés (en utilisant la table `Installer`).
9. Pour chaque salle, le nombre de postes (à partir de la table `Poste`).
10. Pour chaque logiciel, le nombre d'installations sur des postes différents.
11. Moyenne des prix des logiciels 'Unix'.
12. Plus récente date d'achat d'un logiciel.
13. Numéros des postes hébergeant 2 logiciels.

14. Nombre de postes hébergeant 2 logiciels (utiliser la requête précédente en faisant un `SELECT` dans la clause `FROM`).

Exercice 44 : Requêtes multitables

Opérateurs ensemblistes

15. Types de postes non recensés dans le parc informatique (utiliser la table `Types`).
16. Types existant à la fois comme types de postes et de logiciels.
17. Types de postes de travail n'étant pas des types de logiciel.

Jointures procédurales

18. Adresses IP des postes qui hébergent le logiciel 'log6'.
19. Adresses IP des postes qui hébergent le logiciel de nom 'Oracle 8'.
20. Noms des segments possédant exactement trois postes de travail de type 'TX'.
21. Noms des salles où l'on peut trouver au moins un poste hébergeant le logiciel 'Oracle 6'.
22. Nom du logiciel acheté le plus récent (utiliser la requête 12).

Jointures relationnelles

Écrire les requêtes 18, 19, 20, 21 avec des jointures de la forme relationnelle. Numéroter ces nouvelles requêtes de 23 à 26.

27. Installations (nom segment, nom salle, adresse IP complète, nom logiciel, date d'installation) triées par segment, salle et adresse IP.

Jointures SQL2

Écrire les requêtes 18, 19, 20, 21 avec des jointures SQL2 (JOIN, NATURAL JOIN, JOIN USING). Numéroter ces nouvelles requêtes de 28 à 31.

Exercice 45 : Modifications synchronisées

Écrivez le script `modifSynchronisées.sql` pour ajouter les lignes suivantes dans la table `Installer` :

Installer

nPoste	nLog	numIns	dateIns	delai
...	...	...	...	...
p2	log6	séquence...	SYSDATE	NULL
p8	log1		SYSDATE	NULL
p10	log1		SYSDATE	NULL

Figure 12 : Lignes à ajouter

Écrivez les requêtes `UPDATE` synchronisées de la forme suivante :

```
UPDATE table1 alias1
  SET colonne = (SELECT COUNT(*)
                 FROM  table2 alias2
                 WHERE alias2.colonneA = alias1.colonneB…);
```

Pour mettre à jour automatiquement les colonnes rajoutées :

- `nbSalle` dans la table `Segment` (nombre de salles traversées par le segment) ;
- `nbPoste` dans la table `Segment` (nombre de postes du segment) ;
- `nbInstall` dans la table `Logiciel` (nombre d'installations du logiciel) ;
- `nbLog` dans la table `Poste` (nombre de logiciels installés par poste).

Vérifier le contenu des tables modifiées (`Segment`, `Logiciel` et `Poste`).

Exercice 46 : Opérateurs existentiels

Ajoutez au script `requêtes.sql`, les instructions `SELECT` pour extraire les données suivantes :

Sous-interrogation synchronisée

32. Noms des postes ayant au moins un logiciel commun au poste 'p6' (on doit trouver les postes p2, p8 et p10).

Divisions

33. Noms des postes ayant les mêmes logiciels que le poste 'p6' (les postes peuvent avoir plus de logiciels que 'p6'). On doit trouver les postes 'p2' et 'p8' (division inexacte).
34. Noms des postes ayant exactement les mêmes logiciels que le poste 'p2' (division exacte), on doit trouver 'p8'.

Exercice 47 : Extractions dans la base *Chantiers*

Écrivez dans le script `reqchantier.sql` les requêtes SQL permettant d'extraire :

35. Numéro et nom des conducteurs qui étaient sur la route un jour donné (format *jj/mm/aaaa*).
36. Numéro et nom des passagers qui ont visités un chantier un jour donné (format *jj/mm/aaaa*).
37. En déduire le numéro et nom des employés qui n'ont pas bougés de chez eux le même jour.
38. Numéro des chantiers visités les entre le 2 et le 3 du mois d'une année et d'un mois donné avec le nombre de visites pour chacun d'eux.
39. En déduire les chantiers les plus visités.
40. Nombre de visites des employés (transportés comme conducteur) pour un mois donné.
41. Temps de conduite de chaque conducteur d'un mois donné.
42. Numéro du conducteur qui a fait le plus de kilométrage dans l'année avec le kilométrage total.
43. Nom et qualification du conducteur autorisé à piloter tous les types de véhicule.

Partie 5

Les objectifs de ces exercices sont :

- *de créer des vues monotables et multitables ;*
- *d'insérer des enregistrements dans des vues ;*
- *d'effectuer une mise à jour conditionnée via une vue.*

Exercice 51 : Vues monotables

Vues sans contraintes

Écrivez le script `vues.sql`, permettant de créer :

- La vue `LogicielsUnix` qui contient tous les logiciels de type 'Unix' (toutes les colonnes sont conservées). Vérifier la structure et le contenu de la vue (`DESC` et `SELECT`).
- La vue `Poste_0` de structure (`nPos0`, `nomPoste0`, `nSalle0`, `TypePoste0`, `indIP`, `ad0`) qui contient tous les postes du rez-de-chaussée (`etage=0` au niveau de la table `Segment`). Faire une jointure procédurale sinon la vue sera considérée comme une vue multitable. Vérifier la structure et le contenu de la vue.

Insérez deux nouveaux postes dans la vue, tels qu'un poste soit connecté au segment du rez-dechaussée et l'autre à un segment d'un autre étage. Vérifier le contenu de la vue et celui de la table.

Conclusion ?

Supprimez ces deux enregistrements de la table `Poste`.

Résoudre une requête complexe

Créez la vue `SallePrix` de structure (`nSalle`, `nomSalle`, `nbPoste`, `prixLocation`) qui contient les salles et leur prix de location pour une journée (fonction du nombre de postes). Le montant de la location d'une salle à la journée sera d'abord calculé sur la base de 100 € par poste. Servezvous de l'expression `100*nbPoste` dans la requête de définition.

Vérifiez le contenu de la vue, puis afficher les salles dont le prix de location dépasse 150 €.

Ajoutez la colonne `tarif` de type `NUMBER(3)` à la table `Types`. Mettez à jour cette table de manière à insérer les valeurs suivantes :

Type du poste	Tarif en F
TX	50
PCWS	100
PCNT	120
UNIX	200
NC	80
BeOS	400

Figure 13 : Tarifs des postes

Créez la vue `SalleIntermédiaire` de structure (`nSalle, typePoste, nombre, tarif`), de telle sorte que le contenu de la vue reflète le tarif ajusté des salles en fonction du nombre et du type des postes de travail. Il s'agit de grouper par salle, type et tarif (tout en faisant une jointure avec la table Types pour les tarifs), et de compter le nombre de postes pour avoir le résultat suivant :

```
NSALLE  TYPEPOSTE      NOMBRE       TARIF
------- --------- ---------- ----------
s01     TX                 2          50
s01     UNIX               1         200
s02     PCWS               2         100
…
```

À partir de la vue `SalleIntermédiaire`, créez la vue `SallePrixTotal`(`nSalle`, `PrixRéel`) qui reflète le prix réel de chaque salle (par exemple la s01 sera facturée 2*50 + 1*200 = 300€). Vérifiez le contenu de cette vue.

Affichez les salles les plus économiques à la location.

Vues avec contraintes

Remplacez la vue `Poste0` en rajoutant l'option de contrôle (`CHECK OPTION`). Tenter d'insérer un poste appartenant à un étage différent du rez-de-chaussée.

Créez la vue `Installer0` de structure (`nPoste`, `nLog`, `dateIns`) ne permettant de travailler qu'avec les postes du rez-de-chaussée, tout en interdisant l'installation d'un logiciel de type 'PCNT'.
Tentez d'insérer deux postes dans cette vue ne correspondant pas à ces deux contraintes : un poste d'un étage, puis un logiciel de type 'PCNT'. Insérer l'enregistrement 'p6', 'log2' qui doit passer à travers la vue.

Exercice 52 : Vue multitable

Créez la vue `SallePoste` de structure (`nomSalle`, `nomPoste`, `adrIP`, `nomTypePoste`) permettant d'extraire toutes les installations sous la forme suivante :

```
NOMSALLE        NOMPOSTE             ADRIP          NOMTYPEPOSTE
--------------- -------------------- ------------   ----------------
Salle 1         Poste 1              130.120.80.01 Terminal X-Window
Salle 1         Poste 2              130.120.80.02 Système Unix
…
```

Exercice 53 : Mises à jour conditionnées

À partir de la table `Vol` ci-dessous, définissez la vue `v_Vols` qui permettra, à l'aide d'une instruction MERGE, de mettre correctement à jour la table Primes.

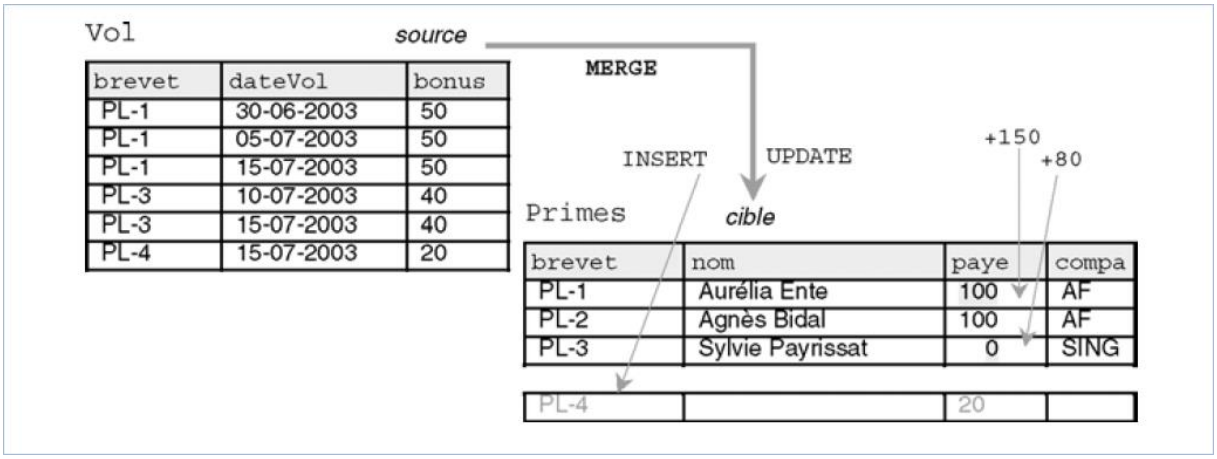

Figure 14 : Mises à jour conditionnées

Exercice 54 : Vues de la base *Chantiers*

Créez la vue `chantier_passagers` permettant d'extraire le détail des visites des employés en tant que passagers d'un mois donné sous la forme suivante (ici pour Avril 2008) :

```
CHANTIER   JOUR       VEHICULE   PASSAGER   CONDUCTEUR      TEMPS
---------- ---------- ---------- ---------- ---------- ----------
CH1        01/04/08   V1         E7         E1                2,5
CH1        01/04/08   V1         E8         E1                2,5
CH1        02/04/08   V2         E1         E10                 2
...
```

Créez la vue `chantier_conducteur` permettant d'extraire le temps passé sur la route par les conducteurs des visites d'un mois donné sous la forme suivante :

```
CHANTIER   CONDUCTEUR JOUR            TEMPS
---------- ---------- ---------- ----------
CH1        E1         01/04/08          2,5
...
```

Créez la vue `chantier_conducteur_passagers` permettant d'extraire le temps passé sur la route par les employés (conducteur ou passager) d'un mois donné sous la forme suivante :

```
EMPLOYE    Temps passé
---------- -----------
E1                 8,5
E2               4,875
...
```

En utilisant ces vues, écrivez la requête qui permet de facturer le temps passé par les employés sur tous les chantiers. La formule à programmer est la suivante : pour tout chantier, le prix est égal au nombre d'employés multiplié par le temps passé (sur la base de 30 euros de l'heure).

Un exemple est donné ci-après :

```
---------- ---------- -------------- ----------
CH1              15,5              7       3255
CH2                 9             11       2970
...
```

Partie 6

L'objectif de ces exercices est d'écrire des blocs PL/SQL puis des transactions PL/SQL manipulant des tables du schéma Parc Informatique.

Exercice 61 : Tableaux et structures de contrôle

Écrivez le bloc PL/SQL qui programme la fusion de deux tableaux (déjà triés par ordre croissant) en un seul (utiliser des structures `WHILE`…). Il faudra afficher ce nouveau tableau (utiliser une structure `FOR`…) ainsi que le nombre d'éléments de ce dernier.

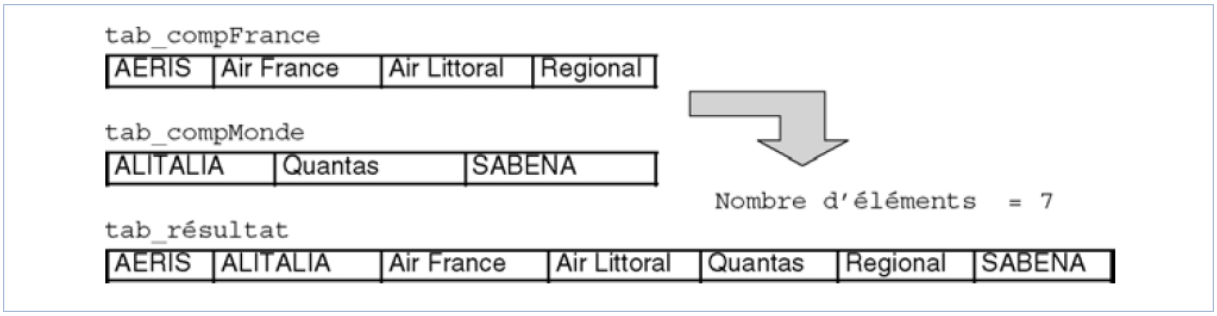

Figure 15 : Fusion de deux tableaux

Exercice 62 : Bloc PL/SQL et variables %TYPE

Écrivez le bloc PL/SQL qui affiche, à l'aide du paquetage `DBMS_OUTPUT`, les détails de la dernière installation sous la forme suivante :

```
Dernière installation en salle : numérodeSalle
-----------------------------------------
Poste : numérodePoste Logiciel : nomduLogiciel en date du dateInstallation
```

Vous utiliserez les directives `%TYPE` pour extraire directement les types des colonnes et pour améliorer ainsi la maintenance du bloc.

Ne tenez pas compte, pour le moment, des erreurs qui pourraient éventuellement se produire (aucune installation de logiciel, poste ou logiciel non référencés dans la base, etc.).

Exercice 63 : Variables de substitution et globales

Écrivez le bloc PL/SQL qui saisit un numéro de salle et un type de poste, et qui retourne un message indiquant les nombres de postes et d'installations de logiciels correspondantes sous la forme suivante :

```
Numéro de Salle : numérodeSalle
Type de poste : typedePoste

G_NBPOSTE
----------
nombredePostes
G_NBINSTALL
-----------
nombred'installations
```

Vous utiliserez des variables de substitution pour la saisie et des variables globales pour les résultats. Vous exécuterez le bloc à l'aide de la commande `start` et non pas par copier-coller (à cause des ordres `ACCEPT`). Ne tenez pas compte pour le moment d'éventuelles erreurs (aucun poste trouvé ou aucune installation réalisée, etc.).

Exercice 64 : Transaction

Écrivez une transaction permettant d'insérer un nouveau logiciel dans la base après avoir saisi toutes ses caractéristiques (numéro, nom, version et type du logiciel). La date d'achat doit être celle du jour. Tracer avec `PUT_LINE` l'insertion du logiciel (message `Logiciel` inséré dans la base). Il faut ensuite procéder à l'installation de ce logiciel sur le poste de numéro 'p7' (utiliser une variable pour pouvoir plus facilement modifier ce paramètre). L'installation doit se faire à la date du jour. Pensez à actualiser correctement la colonne `delai` qui mesure le délai (`INTERVAL`) entre l'achat et l'installation.

Pour ne pas que ce délai soit nul (les deux insertions se font dans la même seconde dans cette transaction), placer une attente de 5 secondes entre les insertions avec l'instruction `DBMS_LOCK.SLEEP(5);`. Utiliser la fonction `NUMTODSINTERVAL` pour calculer ce délai. Tracer avec `PUT_LINE` l'insertion de l'installation. La trace suivante donne un exemple de ce que vous devez produire (les champs en gras sont ceux saisis) :

```
SQL> start exo3plsql
Numéro de logiciel : log15
Nom du logiciel    : Oracle Web Agent
Version du logiciel : 15.5
Type du logiciel    : Unix
Prix du logiciel (en euros) : 1500
Logiciel inséré dans la base
Date achat : 17-07-2003 13:48:08
Date installation : 17-07-2003 13:48:13
Logiciel installé sur le poste

Procédure PL/SQL terminée avec succès.
```

Attente de 5 secondes à ce niveau

Vérifiez l'état des tables mises à jour après la transaction. Ne tenez pas compte pour le moment d'éventuelles erreurs (numéro du logiciel déjà référencé, type du logiciel incorrect, installation déjà réalisée, etc.).

Partie 7

L'objectif de ces exercices est d'écrire des déclencheurs et des sous-programmes PL/SQL manipulant des curseurs et gérant des exceptions sur les bases de données Parc informatique et Chantiers.

Exercice 71 : Curseur

On désire connaître, pour chaque logiciel installé, le temps (nombre de jours) passé entre l'achat et l'installation. Ce calcul devra renseigner la colonne `delai` de la table `Installer` pour l'instant nulle. Les résultats devront être affichés (par `DBMS_OUTPUT.PUT_LINE`) ainsi que les incohérences (date d'installation antérieure à la date d'achat, date d'installation ou date d'achat inconnue).

Écrivez la procédure `calculTemps` pour programmer ce processus. Un exemple d'état de sortie est présenté ci-après :

```
Logiciel Oracle 6 sur Poste 2, attente 2924 jour(s).
Logiciel Oracle 8 sur Poste 2, attente 1463 jour(s).
Date d'achat inconnue pour le logiciel SQL*Net sur Poste 2
Pas de date d'installation pour le logiciel WinDev sur Poste 4
…
Logiciel I. I. S. installé sur Poste 7 11 jour(s) avant d'être acheté!
…
```

Exercice 72 : Transaction

Écrivez la procédure `installLogSeg` permettant d'effectuer une installation groupée sur tous les postes d'un même segment d'un nouveau logiciel. La transaction doit enregistrer dans un premier temps le nouveau logiciel puis, les différentes installations sur tous les postes du segment de même type que celui du logiciel acheté. L'installation se fera à la date du jour.

Ne pas encore tenir compte des éventuelles exceptions et tracer les insertions. Utiliser les paramètres suivants pour tester votre procédure :

```
SQL> EXECUTE installLogSeg('130.120.80', 'log99','Blaster', '05-09-
2003', '9.9', 'PCWS', 999.9 )
Blaster stocké dans la table Logiciel
Installation sur Poste 4 dans Salle 2
Installation sur Poste 5 dans Salle 2

Procédure PL/SQL terminée avec succès.
```

Exercice 73 : Exceptions

Modifiez la procédure `installLogSeg` afin de prendre en compte les exceptions potentielles :

- numéro de segment inconnu (erreur prédéfinie `NO_DATA_FOUND`) ;
- numéro de logiciel déjà présent (erreur prédéfinie `DUP_VAL_ON_INDEX`) ;
- date d'achat plus grande que celle du jour (erreur utilisateur `date_fausse`) ;
- type de logiciel inconnu (erreur non prédéfinie de code Oracle -2291) ;
- aucune installation réalisée, car pas de poste de travail de ce type (erreur utilisateur `pas_install_possible`).

Testez chacun de ces cas avec les valeurs suivantes :

```
--Mauvais segment
EXECUTE installLogSeg('130.120.87', … )
--Logiciel déjà présent
EXECUTE installLogSeg('130.120.80', 'log6',….)
--date > jour
EXECUTE installLogSeg('130.120.80', 'log66', 'Test', '05-09-3000', …)
--Type de logiciel inconnu
EXECUTE installLogSeg('130.120.80','log66','Test','05-09-2003','9.9',
'APPL', …)
--Aucune install
EXECUTE installLogSeg('130.120.81', 'log55', '…', '…', '…', 'PCWS', …)
--Bonne installation
EXECUTE installLogSeg('130.120.80', 'log66','Eudora6', '10-09-2003',
'6.0', 'PCWS', 66)
```

Exercice 74 : Déclencheurs

Mises à jour de colonnes

Écrivez le déclencheur `Trig_Après_DI_Installer` sur la table `Installer` permettant de faire la mise à jour automatique des colonnes `nbLog` de la table `Poste`, et `nbInstall` de la table `Logiciel`. Prévoir les cas de désinstallation d'un logiciel sur un poste, et d'installation d'un logiciel sur un autre. Écrivez le déclencheur `Trig_Après_DI_Poste` sur la table `Poste` permettant de mettre à jour la colonne `nbPoste` de la table `Salle` à chaque ajout ou suppression d'un nouveau poste. Écrivez le déclencheur `Trig_Après_U_Salle` sur la table `Salle` qui met à jour automatiquement la colonne `nbPoste` de la table `Segment` après la modification de la colonne `nbPoste`. Ces deux derniers déclencheurs vont s'enchaîner : l'ajout ou la suppression d'un poste entraînera l'actualisation de la colonne `nbPoste` de la table `Salle` qui conduira à la mise à jour de la colonne `nbPoste` de la table `Segment`. Ajouter un poste pour vérifier le rafraîchissement des deux tables (`Salle` et `Segment`). Supprimer ce poste puis vérifier à nouveau la cohérence des deux tables.

Programmation de contraintes

Écrivez le déclencheur `Trig_Avant_UI_Installer` sur la table `Installer` permettant de contrôler, à chaque installation d'un logiciel sur un poste, que le type du logiciel correspond au type du poste, et que la date d'installation est soit nulle soit postérieure à la date d'achat.

Exercice 75 : Transaction de la base *Chantiers*

Écrivez la procédure `finAnnee` permettant de rajouter à chaque véhicule les kilométrages faits lors des visites de l'année. Vous utiliserez un seul curseur pour parcourir tous les véhicules. Il faudra ensuite supprimer toutes les missions de l'année (visites et détails des trajets des employés transportés).

Exercice 76 : Déclencheurs de la base *Chantiers*

Déclencheur ligne

Écrivez le déclencheur `TrigPassagerConducteur` sur la table `transporter` permettant de vérifier qu'à chaque nouveau transport, le passager déclaré n'est pas déjà enregistré en tant que conducteur le même jour.

Déclencheur composé

Écrivez le déclencheur composé `TrigcapaciteVehicule` sur la table `transporter` permettant de contrôler, qu'à chaque nouveau transport, la capacité du véhicule n'est pas dépassée.

Vous éviterez le problème des tables mutantes en :

- déclarant dans la zone de définition commune un tableau recensant le nombre de personnes transportées par visite ;
- déclarant dans cette même zone un curseur qui va parcourir toutes les visites ;
- chargeant le tableau dans la section `BEFORE STATEMENT` ;
- examinant le tableau dans la section `BEFORE EACH ROW` et en le comparant avec les données à insérer.

Les messages à afficher pour tracer et rendre plus lisible ce déclencheur sont :

- dans la section `BEFORE EACH ROW` : "Enregistrement du transport de nom" puis éventuellement " Premier trajet de la visite" ;
- dans la section `AFTER EACH ROW` : "Transport de nom bien enregistré" puis "Il ne reste plus que x place(s) disponible(s)" ;
- dans la section `AFTER STATEMENT` : "Nombre de trajet(s) traité(s) : nombre" ;

Les messages d'erreur à produire le cas échéant sont les suivants :

- "Capacité max atteinte n pour la visite chantier du date, pour le véhicule v" ;
- "BASE INCORRECTE : Capacité dépassée n pour la visite chantier du date, pour le véhicule v".