CDI : Contraintes et domaine d'intégrité
Ventes d'une chaîne de magasin

Objectifs
Procédures stockées ; Utilisation du langage **PL/SQL**
Contraintes statiques et dynamiques (déclencheurs)
Chargement des données
SQL pour Oracle

Section
BUT Informatique de CAEN (Campus 3)
- R2.06 : Exploitation d'une base de données
- R3.07 : SQL et programmation

Auteurs
P. Monchy maj par E.Porcq

Références
Oracle PL/SQL par SheikYerbouti (pl/sql.pdf)
Les déclencheurs (E.Porcq)
Intégrité des données (P.Monchy maj E.Porcq)

1 Objectif de l'étude

La base de données étudiées présente l'intérêt d'être très mal conçue.
- Liens très (trop) nombreux entre les tables
- Propriétés dupliquées
- Pas de contraintes (contraintes de clés, contraintes d'intégrité, contraintes de domaines)
- Données manquantes
- Enregistrements dupliqués

L'objectif est donc d'apprendre à utiliser des techniques pour améliorer une base en production.

2 Introduction

L'exemple suivant s'appuie sur une base de données relative aux ventes d'une chaîne de magasins. Les magasins partagent un stock commun et bénéficient d'une gestion centralisée des commandes et des livraisons.

Le modèle de gestion est loin d'être complet, car il n'est pas prévu de gérer les commandes auprès des fournisseurs, ni la facturation, ni le suivi du paiement des clients.

2.1 Le modèle entités-associations

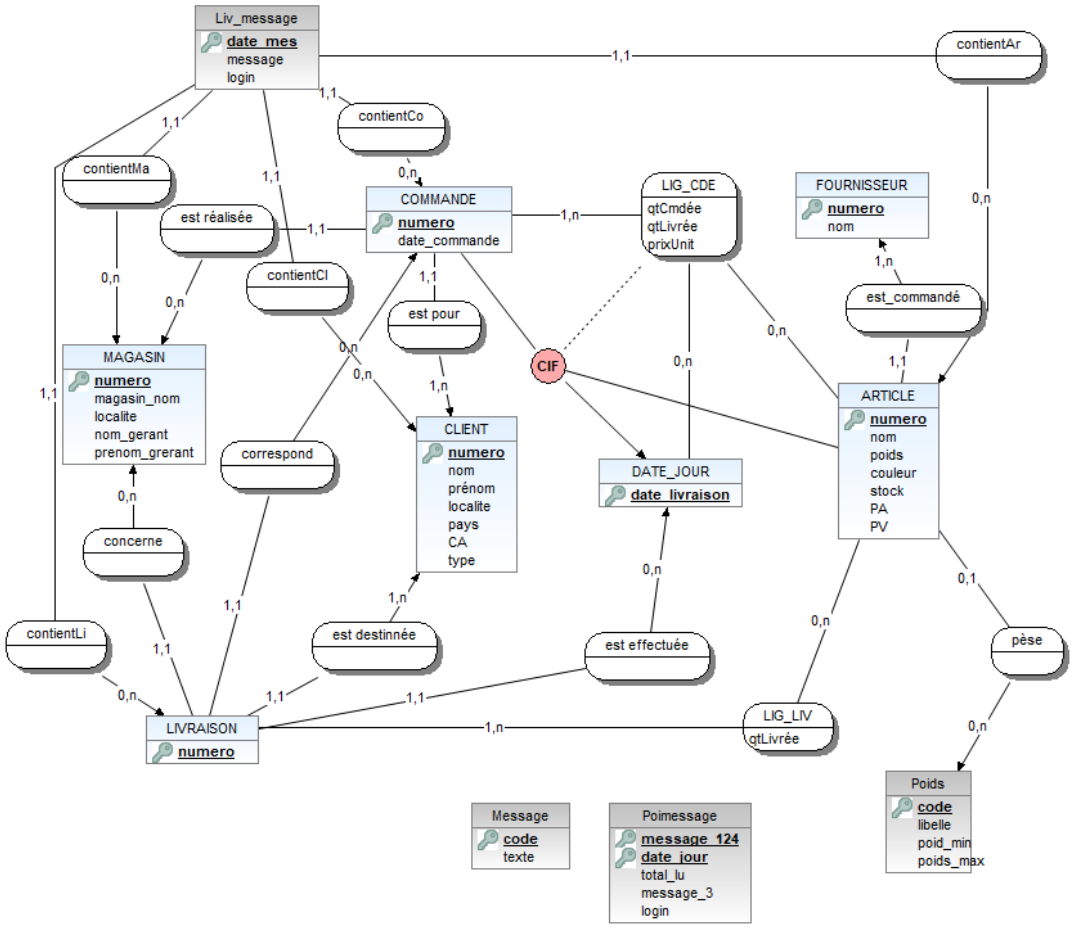


Figure 21a : Modèle Conceptuel de données

- Un article est associé à un fournisseur,
- une commande concerne un client,
- un magasin et est constituée de plusieurs lignes de commande,
- chacune des lignes de commande concerne un article,
- une commande peut être livrée en plusieurs fois,
- chaque livraison est constituée de plusieurs lignes de livraison,
- chaque ligne de livraison concerne un article et correspond toujours à une seule ligne de commande.

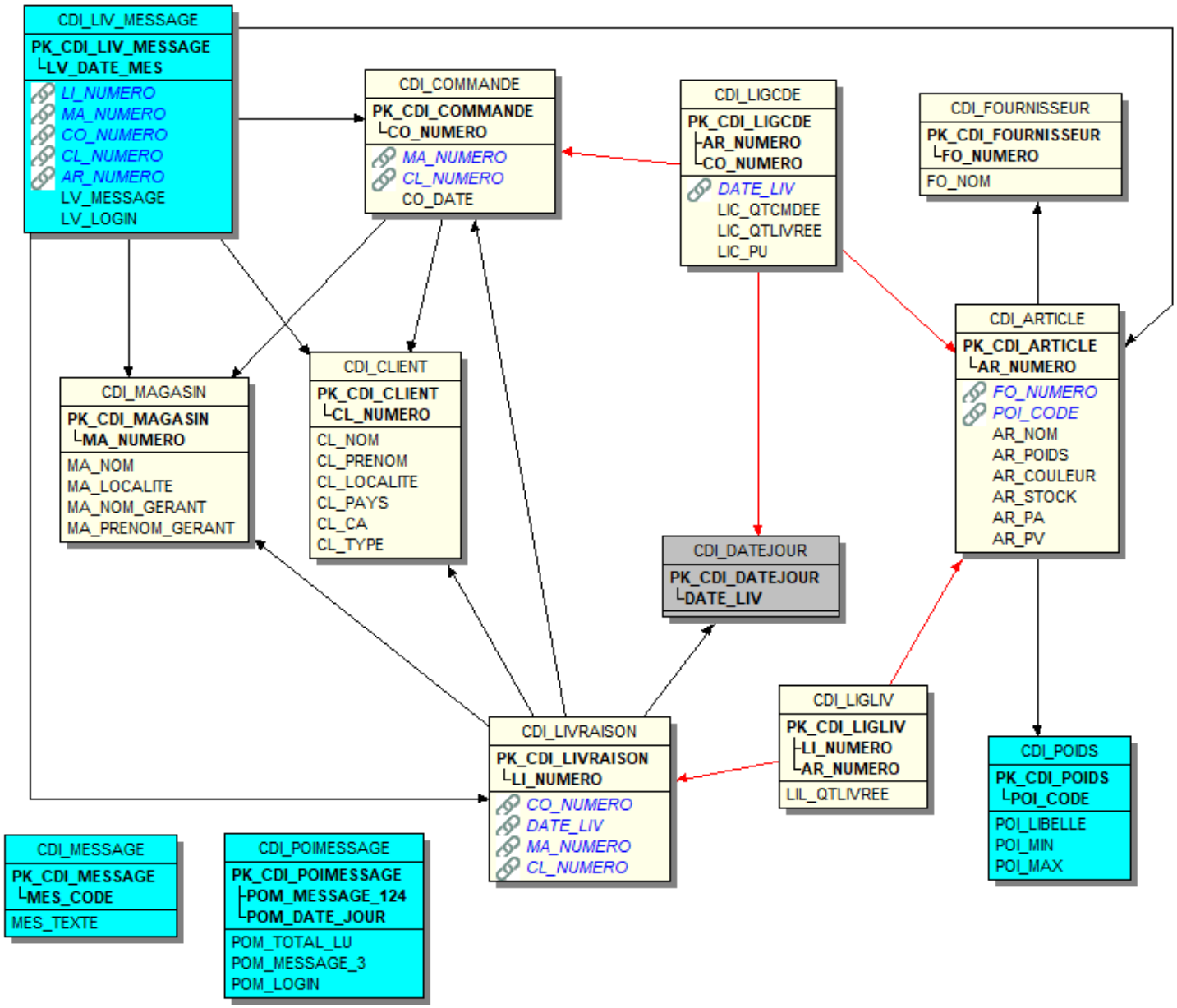


Figure 21b : Modèle Logique de données

CDI_CLIENT(CL_NUMERO,CL_NOM,CL_PRENOM ,CL_LOCALITE ,CL_PAYS,CL_CA,CL_TYPE)
CDI_MAGASIN(MA_NUMERO,MA_NOM, MA_LOCALITE,MA_NOM_GERANT,MA_PRENOM_GERANT)
CDI_FOURNISSEUR(FO_NUMERO,FO_NOM)
CDI_COMMANDE(CO_NUMERO,CO_DATE,#MA_NUMERO,#CL_NUMERO)
CDI_ARTICLE(AR_NUMERO,#FO_NUMERO,AR_NOM,AR_POIDS,AR_COULEUR,AR_STOCK,AR_PA,AR_PV)
CDI_LIVRAISON(LI_NUMERO,#MA_NUMERO,#CO_NUMERO,#CL_NUMERO,DATE_LIV)
CDI_LIG_CDE(#CO_NUMERO,#AR_NUMERO,LIC_QTCMDEE,LIC_QTLIVREE,LIC_PU,DATE_LIV)
CDI_LIGLIV(#LI_NUMERO,#AR_NUMERO,LIL _QTLIVREE)

Certaines tables seront utilisées uniquement au chapitre PL/SQL

CDI_DATEJOUR n'est pas devenue une table !

2.2 Le schéma des tables

- ✗ Commenter le MCD en insistant sur la CIF.
- ✗ Commenter le Modèle Logique de Données Relationnel (texte ou graphique).

2.3 Le chargement des données

- À faire sur un PC de la 2236 et pas sur portable (à cause du pare-feu)
- Utilisez Windows et pas Linux

À partir du modèle relationnel et de fichiers fournis, créer les tables de la base.

	Création du schéma			Chargement des données				
tables à créer	import	script	sql dev	import	script	SqlLoader	db link	tableur
CDI_Article		■	■					■
CDI_Fournisseur		■	■		■			
CDI_Client	■			■				
CDI_Commande	■			■				
CDI_Magasin	■			■				
CDI_LigCde			■				■	
CDI_Livraison			■			■		
CDI_LigLiv			■			■		

2.3.1 Le fichier "***cdi_ArtFour.sql***" contient le script de la table CDI_Article et CDI_Fournisseur.

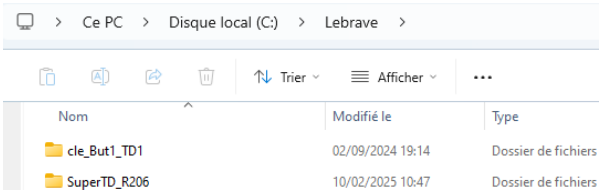
- Les données de "***cdi_fournisseur***" seront chargées à partir du schéma " eric "
- Les données de "***cdi_article***" seront insérées à partir du fichier "***article_etudfiant.ods***" (fabriquer les requêtes d'insertion avec le tableur).

2.3.2 Le fichier "***cdi_clicommag.dmp***" contient le schéma des tables CDI_Client, CDI_Commande et CDI_Magasin. Les tables seront importées avec leurs données. Comparer les tables importées à votre modèle relationnel.

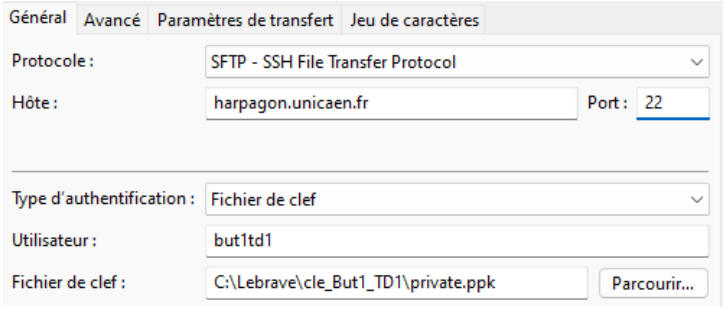
Nous utiliserons un outil d'import (" import ") en ligne de commande présent sur le serveur (distant) Oracle.

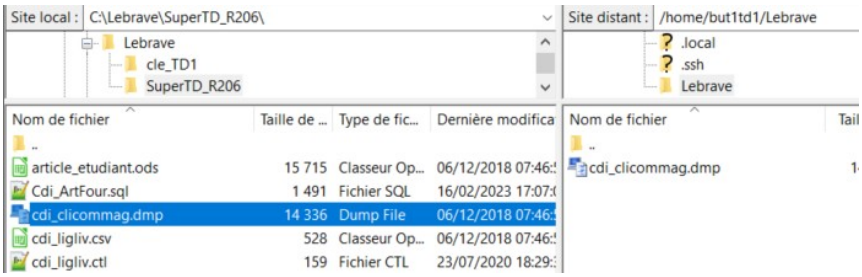
Exemple avec l'étudiant Stéphane LEBRAVE (etupass : GraphBien) au compte Oracle etu1_90 du TD1

- Stéphane va se connecter en SFTP sur le serveur Oracle pour y déposer le fichier dmp à importer
  - il a créé sur son pc un dossier avec les fichiers du TD et un dossier avec la clé (un peu fayot ce Lebrave).

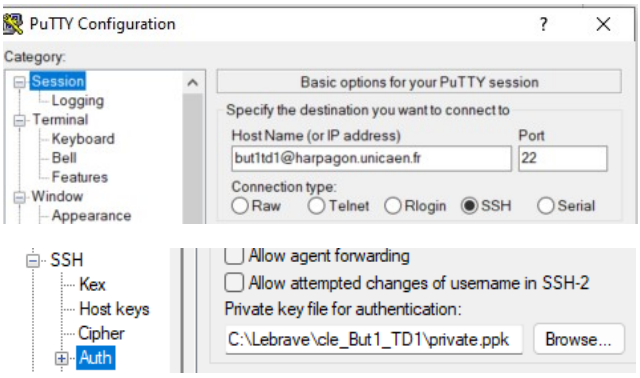


  - SFTP : avec **filezilla**, il se connecte au serveur, crée un dossier à son nom et y dépose le fichier clicommag.dmp





- SSH : Il se connecte en mode console à distance en utilisant **putty**
  - Exemple avec etudiant de TD1 nommé etu1



  - il utilise l'outil d'importation

```
but1td1@kharpagon:~$ cd Lebrave
but1td1@kharpagon:~/Lebrave$
but1td1@kharpagon:~/Lebrave$ source /opt/info-ora.env
but1td1@kharpagon:~/Lebrave$ imp etu1_90
Import: Release 21.0.0.0.0 - Production on Wed Jan 31 16:57:49 2024
Version 21.3.0.0.0
Copyright (c) 1982, 2021, Oracle and/or its affiliates.  All rights reserved.
Password: ETU1_90
Connected to: Oracle Database 21c Enterprise Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
Import data only (yes/no): no >
Import file: expdat.dmp > cdi_clicommag.dmp
Enter insert buffer size (minimum is 8192) 30720>
Export file created by EXPORT:V10.02.01 via conventional path
Warning: the objects were exported by CDI, not by you
import done in US7ASCII character set and AL16UTF16 NCHAR character set
import server uses AL32UTF8 character set (possible charset conversion)
export client uses WE8MSWIN1252 character set (possible charset conversion)
List contents of import file only (yes/no): no >
Ignore create error due to object existence (yes/no): no >
Import grants (yes/no): yes >
Import table data (yes/no): yes >
Import entire export file (yes/no): no > yes
. importing CDI's objects into ETU1_90
. importing CDI's objects into ETU1_90
. . importing table                  "CDI_CLIENT"         16 rows imported
. . importing table                "CDI_COMMANDE"         13 rows imported
. . importing table                 "CDI_MAGASIN"         12 rows imported
Import terminated successfully without warnings.
```

2.3.3 Le schéma de la table " ***cdi_LigCde*** **"** est à créer en langage SQL à partir du compte Oracle " *eric* " dont vous aurez les droits.

2.3.4 Le schéma des tables " ***cdi_Livraison*** " et " ***cdi_LigLiv*** " est à créer en langage SQL (pour la création de ces deux tables, vérifier les types des autres tables). Ces tables seront créées dans un nouveau tablespace.

N'ajouter aucune contrainte.

Les données seront chargées à partir des fichiers de données cdi_ligliv.csv et cdi_livraison.csv en utilisant l'outil ***SqlLoader*** du serveur Oracle ; les fichiers de contrôle sont fournis également.

Il faudra téléverser ces fichiers sur le serveur dans votre dossier.

Comparer le contenu des fichiers csv avec les enregistrements importés dans les tables.

Commande à exécuter en SSH (putty) :

- sqlldr userid=ETU1_XX/ETU1_XX control=cdi_ligliv.ctl
- sqlldr userid=ETU1_XX/ETU1_XX control=cdi_livraison.ctl

Certains messages d'erreurs sont affichés à l'exécution. D'autres erreurs sont lisibles dans le fichier " log" .Il faudra corriger le fichier ctl ou csv en fonction de l'erreur.

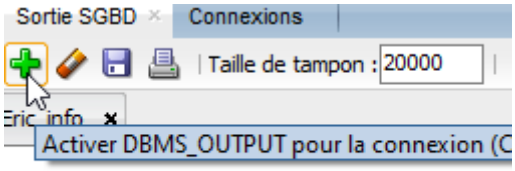
Exemple :

```
but1td1@kharpagon:~/Lebrave$ sqlldr userid=ETU1_90/ETU1_90 control=cdi_ligliv.ctl

SQL*Loader: Release 21.0.0.0.0 - Production on Wed Jan 31 17:44:32 2024
Version 21.3.0.0.0
Copyright (c) 1982, 2021, Oracle and/or its affiliates.  All rights reserved.
Path used:      Conventional
SQL*Loader-466: Column LIL_QTELIVREE does not exist in table CDI_LIGLIV.
```

3 Les procédures

Note : Pour afficher le résultat des dbms_output.put_line sur sqldeveloper, il faut effectuer ceci :

- Menu Affichage → Sortie SGBD
- cliquer sur le ✚



Bien garder le nom des procédures proposées

3.1 Exercices de bases

3.1.1 Tester le bloc pl/sql suivant

```
declare
chaine varchar2(20) := 'salut mon campus ';
begin
dbms_output.put_line( chaine ) ;
dbms_output.put_line( 'nous sommes ' || chr(10) ||' le '|| sysdate  ) ;
end;
/
```

3.1.2 Tester le bloc pl/sql suivant. Donner le rôle de chaque ligne de code.

```
accept num prompt 'veuillez taper un mot : '
set verify off
declare
val varchar2(20);
begin
val := '&num';
dbms_output.put_line( 'vous avez tapé : ' || val) ;
dbms_output.put_line( 'nous sommes le ' || sysdate || chr(10) ) ;
end;
/
```

3.1.3 PLSQL_PROC_313.

Avec SQL developer , créer la table suivante.

```
drop table bidon;
create table bidon
(
 num number(4) primary key,
 type char(10),
 valeur number(4,2)
);
```

Cliquer avec le bouton droit sur procédure, puis sélectionner " new procedure ". La nommer puis saisir le code ci-dessous.

```
CREATE OR REPLACE PROCEDURE PLSQL_PROC_313 AS
BEGIN
 dbms_output.put_line('Ceci est un'||chr(10)||'message');
END PLSQL_PROC_313;
/
```

Compiler, exécuter et commenter le programme.

```
begin
PLSQL_PROC_313;
end;
/
```

3.1.4 PLSQL_PROC_314

Tester (et commenter) le programme suivant :

```
CREATE OR REPLACE
PROCEDURE PLSQL_PROC_314 AS
vNbLignes int := 0;
BEGIN
```

```
 insert into bidon values(3,'ville',35.4);
 insert into bidon values(4,'daine',30.4);
 select count(*) into vNbLignes from bidon;
 dbms_output.put_line('nb réponses : '|| vNbLignes);
END PLSQL_PROC_314;
/

begin
 delete from bidon;
 PLSQL_PROC_314;
end;
/
```

Pourquoi le delete est indispensable ici ?

3.2 Exercices avec structures de contrôle

3.2.1 PLSQL_PROC_321

L'objectif sera d'écrire une procédure permettant de générer les 10 premiers nombres. Utiliser une boucle for

Schéma de la table PLSQL_TAB (à créer)

- nom_exercice varchar2(15)
- date_jour date
- multiplicande number(2)
- multiplicateur number(2)
- resultat number(3) (PK)

Code PLSQL (à compléter)

```
CREATE OR REPLACE
PROCEDURE PLSQL_PROC_321 as
 v_nom_exercice -- à compléter
BEGIN
-- à compléter
END;
```

Diagramme d'activité

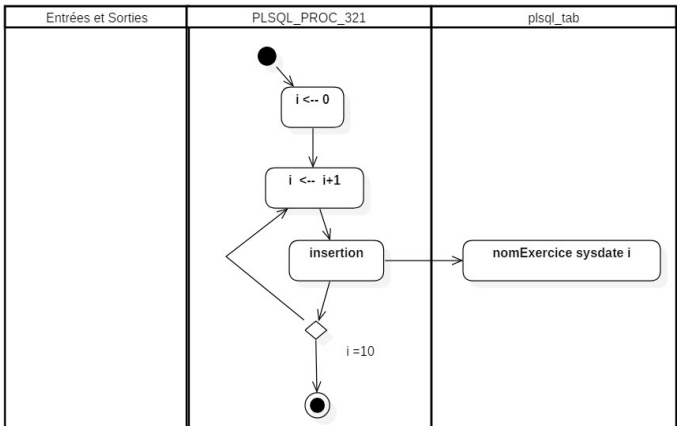


figure 3.2.1

Travail à réaliser

Créer la table PLSQL_TAB.
Écrire la procédure correspondant au diagramme d'activité fourni.
Lancer le programme à partir de sqldeveloppeur puis vérifier le résultat
Lancer une nouvelle fois la procédure. Que se passe-t-il ? Justifier.

Résultat

En SQL : `select * from plsql_tab;`

NOM_EXERCICE	DATE_JOUR	MULTIPLICATEUR	MULTIPLICANDE	RESULTAT
PLSQL_PROC_321	11/03/25	(null)	(null)	1
PLSQL_PROC_321	11/03/25	(null)	(null)	2
PLSQL_PROC_321	11/03/25	(null)	(null)	3
PLSQL_PROC_321	11/03/25	(null)	(null)	4
PLSQL_PROC_321	11/03/25	(null)	(null)	5

3.2.2 PLSQL_PROC_322

Même exercice que précédemment avec une boucle while.

Exécuter le programme au pas par pas.

3.2.3 PLSQL_PROC_323

Même exercice que précédemment mais en affichant à l'écran les données insérées. Utiliser loop … exit

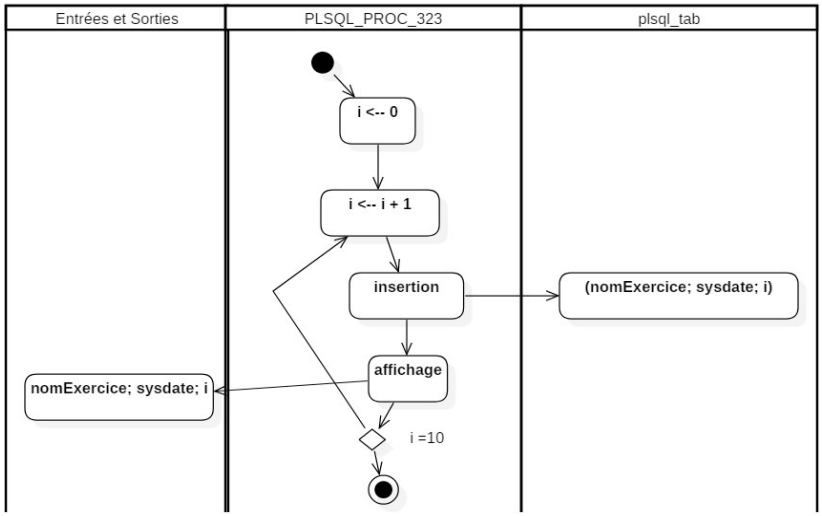


figure 3.2.3

3.3 Exercices avec gestion des erreurs

3.3.1 PLSQL_PROC_331

Objectif : Écrire une procédure stockée permettant de générer une table de multiplication. Le choix du multiplicateur, compris entre 1 et 9 sera saisi au clavier. La table de multiplication sera enregistrée dans la table PLSQL_TAB et affichée à l'écran. Le multiplicande (de 1 à 10) sera généré par une boucle dans le programme.

Tant que la procédure ne fonctionne pas, on ne s'intéresse pas à la saisie.

Diagramme d'activité

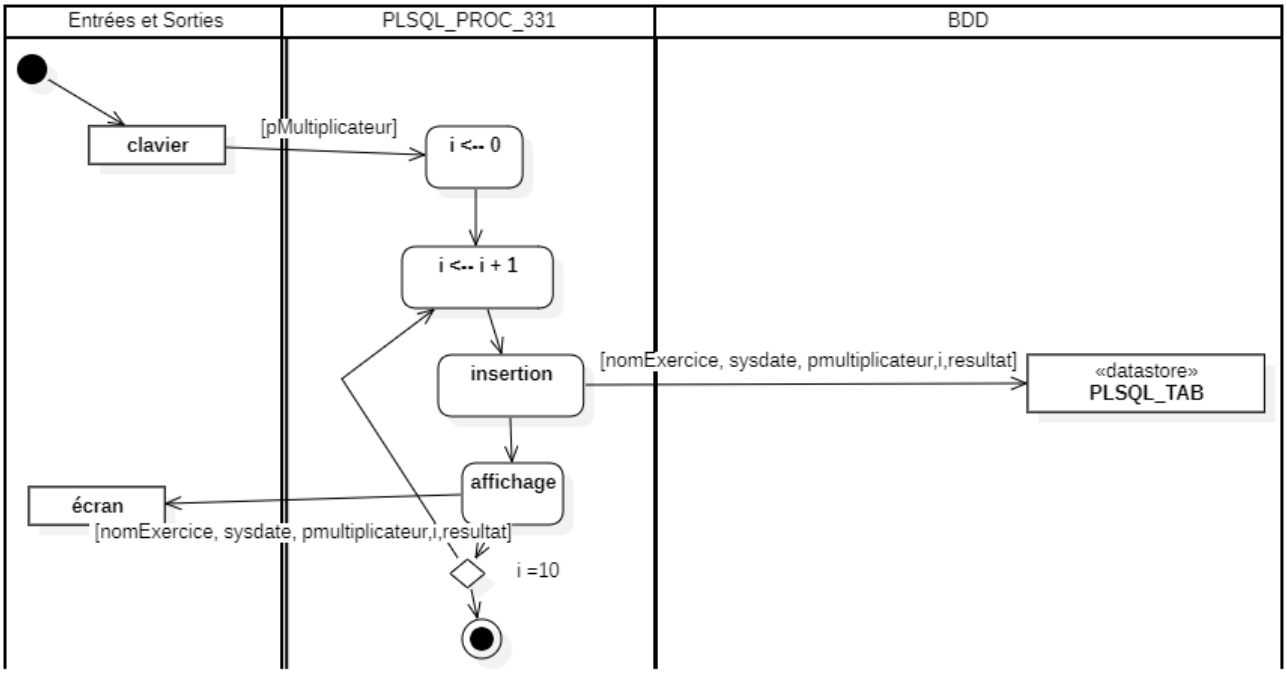


figure 3.3.1

Code PLSQL

```
CREATE OR REPLACE PROCEDURE PLSQL_PROC_331(pmultiplicateur varchar2) AS
-- à compléter
BEGIN
```

```
-- à compléter
END PLSQL_PROC_331;
```

3.3.2 PLSQL_PROC_332

Même exercice que précédemment mais en y ajoutant un contrôle d'anomalie sur la valeur saisie au clavier (hors procédure).

```
create or replace PROCEDURE PLSQL_PROC_332(pMultiplicateur varchar2) AS
vNomExercice plsql_tab.nom_exercice%type := 'PLSQL_PROC_332';
vErreur number(1) :=0;
cMessage1 constant varchar2(100) := 'la valeur saisie ne peut être nulle';
cMessage2 constant varchar2(100) := 'il faut saisir un nombre';
cMessage3 constant varchar2(100) := 'la valeur doit être comprise entre 0 et 10';
BEGIN
 if pMultiplicateur is null then
          à compléter
 else
  for i in 1 .. length(pMultiplicateur)
  loop
   if substr(pMultiplicateur,i,1) not between '0' and '9' then
    dbms_output.put_line(pMultiplicateur || cMessage2);
    vErreur := 1;
   à compléter
```

Commenter le code if substr(pMultiplicateur,i,1) not between '0' and '9'

3.3.3 PLSQL_PROC_333

Même exercice que précédemment mais les erreurs seront traitées dans le secteur Exception de la procédure stockée. Le message affiché sera un message d'erreur.

Instructions clés :

- Exception
- Raise ...
- raise_application_error
- VALUE_ERROR

3.4 Exercices avec curseurs

On imagine pour ces exercices de bases que l'on a recueilli dans une table **sondage** des informations sur des individus. On mémorise, un numéro (unique), la date de naissance de la personne sondée, la réponse à une première question, la réponse à une seconde question puis une valeur correspondant à la note que donne la personne sondée sur un élément donné.

Dans un premier temps, les données ont été enregistrées sans véritables règles (sans contraintes). Nous allons remédier à tout cela.

3.4.1 Construire une table sondage de cette façon :

```
create table sondage
(
 num number(3),
 date_naissance date,
 reponse1 char(20),
 reponse2 varchar2(20),
 val int
);
-- il y a mélange entre des données Oracle et des données Standards
```

num	date_naissance	reponse1	reponse2
1	12/09/2000	oui	un peu
2	13/06/2004	non	beaucoup
3	01/04/1956	o	non
25	16/02/1986	n	pas du tout
26	06/02/1947	oui	à la folie

3.4.2 PLSQL_PROC_342

Insérer une dizaine de lignes dans sondage. Il en faut au moins 3 avec val=7 et 3 dont date_naissance='01/03/2000'.

Ne pas oublier le commit.

```
insert all
into sondage (num,date_naissance,reponse1,reponse2,val) values (26 , to_date('01/01/2000','dd/mm/yyyy') , 'oui' , 'un peu'            , 6 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (29 , to_date('01/05/1991','dd/mm/yyyy') , 'oui' , 'un peu'            , 7 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (38 , to_date('01/06/1980','dd/mm/yyyy') , 'non' , 'trop peu ou trop pas' , 3 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (39 , to_date('01/06/1980','dd/mm/yyyy') , 'non' , 'un peu'            , 3 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (40 , to_date('01/06/1980','dd/mm/yyyy') , 'non' , 'beaucoup'          , 7 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (41 , to_date('01/03/2000','dd/mm/yyyy') , 'non' , 'pas trop'          , 8 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (42 , to_date('01/03/2000','dd/mm/yyyy') , 'non' , 'trop pas'          , 7 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (43 , to_date('01/06/1980','dd/mm/yyyy') , 'non' , 'pas du tout'       , 7 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (45 , to_date('01/06/1980','dd/mm/yyyy') , 'non' , 'trop pas du tout'   , 5 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (46 , to_date('01/03/2000','dd/mm/yyyy') , 'non' , 'trop pas du tout'   , 8 )
into sondage (num,date_naissance,reponse1,reponse2,val) values (47 , to_date('01/03/2000','dd/mm/yyyy') , 'non' , 'trop pas du tout'   , 7 )
SELECT * FROM dual;
commit;
```

Afficher le contenu de sondage.

Afficher les num de sondage dont val = 7.
Afficher les réponses2 de sondage dont la date est '01/03/2000'
Écrire une procédure stockée affichant le nombre d'enregistrements dans la table sondage. **Aucun curseur n'est exigé.**

3.4.3 PLSQL_Exo_343

Créer une procédure stockée (new procedure). En utilisant un curseur, récupérer les numéros d'enregistrements de sondage ainsi que reponse1 pour lesquels val=7 (il faut plusieurs réponses).

L'affichage donnera :
1 : num <valeur>, reponse1 <valeur>
2 : num <valeur>, reponse1 <valeur>
3 : num <valeur>, reponse1 <valeur>
etc ...

Exemple :

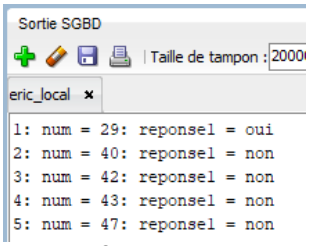


figure 3.4.3

Algorithme de haut niveau
Début
déclarer variables et curseur;
ouvrir curseur
faire
lire donnée du curseur
afficher résultat
tant qu'il reste des données
Afficher ("fin")
fermer curseur
fin

3.4.4 PLSQL_PROC_344

Après avoir saisi une date de naissance au format dd/mm/yyyy, appeler un bloc pl/sql ou une procédure stockée qui devra afficher les " num " des lignes correspondantes (date_naissance dans sondage). Pour ne pas cumuler les difficultés, on se contentera dans un premier temps de passer en dur une date en paramètre de la procédure (pDateNais).

3 exceptions sont prévues :

- pDateNais est vide
- pDateNais n'est pas une date
- pDateNais n'est pas dans la table

a) Version simple :

On copie le pDateNais dans une variable de type date (var_date). Si pDateNais n'est pas au bon format, l'instruction va échouer et provoquer une exception.
Dans le bloc exception, on se contente d'afficher les codes d'erreur pour les dates.

```
EXCEPTION
  when erreurVide then ...;
  when erreurDatePasDansSondage then ...;
  when others then
    dbms_output.put_line(to_char(sqlcode)||':'||to_char(SQLERRM));
```

Tester avec plusieurs erreurs différentes

```
BEGIN
  -- PLSQL_PROC_344A('01/06/1980');
  -- PLSQL_PROC_344A('01/03/2022');
  --  PLSQL_PROC_344A('15/06/-05');
  --  PLSQL_PROC_344A('36/06/2020');
  -- PLSQL_PROC_344A('25');
  PLSQL_PROC_344A('toto');
END;
/
```

b) Version complète :

Décoder les erreurs et afficher un message spécifique :
Exception
when others then

```
dbms_output.put_line(to_char(sqlcode));
if to_char(sqlcode) = '-1858' then
  raise_application_error(-20003,'ce n''est pas une date, c''est un texte');
elsif to_char(sqlcode) = '-1840' then
  raise_application_error(-20004,'ce n''est pas une date, c''est un nombre')
elsif ...
```

Exemples de codes d'erreurs

-1830:ORA-01830: Le modèle du format de date se termine avant la conversion de la chaîne d'entrée entière
-1839:ORA-01839: le quantième n'est pas valide pour le mois indiqué
-1840:ORA-01840: valeur entrée pas assez longue pour le format de la date
-1841:ORA-01841: L'année (complète) doit être comprise entre -4713 et +9999 et être différente de 0
-1843:ORA-01843: ce n'est pas un mois valide
-1847:ORA-01847: le jour du mois doit être compris entre 1 et le dernier jour du mois
-1858:ORA-01858: Caractère non numérique trouvé à la place d'un caractère numérique
-1861:ORA-01861: le littéral ne concorde pas avec le format chaîne de caractères

Aide :

Oracle utilise des codes d'exceptions relatifs aux différents événements ou erreurs détectées. En PL/SQL, la variable sqlcode contient le code de l'exception. Pour certains d'entre eux, une constante plus lisible a été créée (TOO_MANY_ROWS, NO_DATA_FOUND …). Pour d'autre, nous n'avons accès qu'au code de l'erreur. Voici une liste non exhaustive des codes d'erreur :

Exception prédéfinie	Erreur Oracle	Valeur de SQLCODE
ACCESS_INTO_NULL ☺	ORA-06530	-6530
CASE_NOT_FOUND	ORA-06592	-6592
COLLECTION_IS_NULL	ORA-06531	-6531
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
SELF_IS_NULL	ORA-30625	-30625
STORAGE_ERROR	ORA-06500	-6500
SUBSCRIPT_BEYOND_COUNT	ORA-06533	-6533
SUBSCRIPT_OUTSIDE_LIMIT	ORA-06532	-6532
SYS_INVALID_ROWID	ORA-01410	-1410
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

figure 3.4.4

Le même bloc " exception " peut traiter les codes oracles et utilisateurs (<-20000)

3.4.5 PLSQL_PROC_345

PLSQL_PROC_345A : Après avoir saisi un mot appeler un bloc pl/sql ou une procédure stockée qui devra afficher les lignes dont la réponse2 contient le paramètre. Aucune gestion d'exception n'est demandée.

Compléter ce programme, le nommer **PLSQL_PROC_345B** qui fait la même fonction mais avec 2 paramètres. Pour cela, il faudra faire en sorte que l'unique curseur reçoive un paramètre. **Ne pas modifier la requête de la version A**. En fait, il faut un curseur recevant un paramètre !

3.5 Exo CDI : Saisie d'un numéro de bon de livraison PLSQL_FUNC_35

Exécuter cdi_etudiant_correct.sql

Contexte :

Notre application permet de garder un journal des opérations effectuées.

Un client se plaint de ne pas avoir été livré. Appel de la **Fonction**_35

- la livraison a-t-elle été prévue ?
  - Oui (M01, PLSQL_FUNC_35 retourne 0). **Contacter le transporteur !**
  - Non (M02, PLSQL_FUNC_35 retourne -1). C'est de notre faute alors. Vérifions davantage (voir 3.6)

L'objectif est donc de chercher les commandes non livrées.

- Un opérateur entre un numéro de livraison.
- Si ce numéro correspond à une livraison en cours ou effectuée, la **fonction** inscrira le message 'M01' (voir table cdi_message) dans une table cdi_liv_message et retournera 0.
- Dans le cas contraire, elle inscrira le message 'M02' et retournera -1.

3 techniques possibles :

- solution 1) la meilleure : PLSQL_FUNC_35A
  - On évite qu'une exception soit levée. Rappel, un select qui lit 0 réponse provoque une exception automatique ***no_data_found***
  - on va plutôt utiliser count pour compter les livraisons du numéro passé en paramètre => pas d'exception
- solution 2) la meilleure. : PLSQL_FUNC_35B
  - On n'évite pas qu'une exception soit levée
  - On tente de lire les infos sur la livraison du numéro passé en paramètre
  - Si exception, on l'intercepte et c'est tout
- solution 3) la moins bonne. On utilise un curseur

Donner le code de cette fonction. Voir diagramme d'activité figure 3.5.

Test :

```
declare
  x int ;
begin
    x:=PLSQL_FUNC_35A('L9719');
    dbms_output.put_line(x);
    dbms_output.put_line('------------');
    x:=PLSQL_FUNC_35A('L9710');
    dbms_output.put_line(x);
end;
/
```

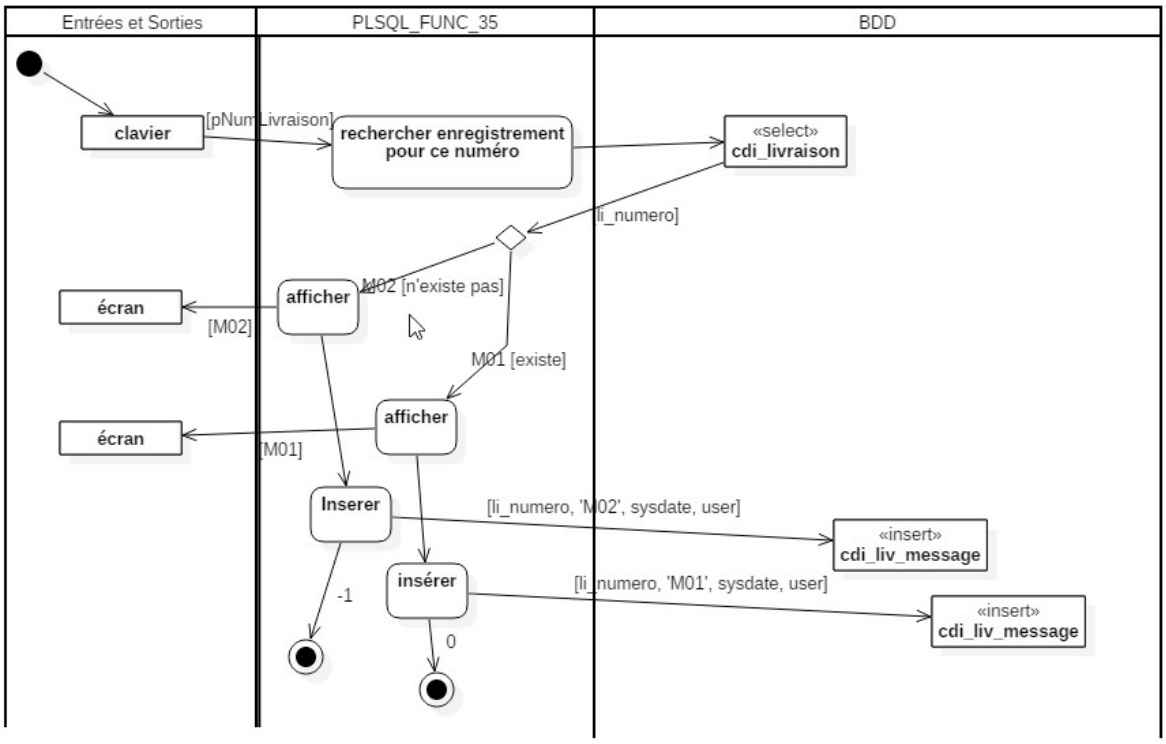


Figure 3.5 : Diagramme d'activité (UML)

3.6 Exo CDI : Saisie d'un numéro de bon de commande PLSQL_PROC_36

On va vérifier la commande en cas de non-livraison.

Proc_36 :

- la livraison a-t-elle été prévue ? => appel de PLSQL_FUNC_35

- oui
- non
  - la commande a-t-elle été faite ?
    - oui (M03) . Il faut que l'on mette en livraison alors !
    - non (M04) : normal que cela ne soit pas livré, cela n'a même pas été commandé. Qui est le coupable ?

L'objectif est de compléter l'exercice précédent par la saisie d'un bon de commande. Le code respectera l'algorithme suivant:

```
Algorithme :
ParamEntrée : pliv_num, pcom_num
début
        res <-- PL_SQL_FUNC_35(pliv_num)
        si res=-1 alors
                vérifier si le co_numero existe
                si v_ vcom_num <> null alors
                        insérer dans cdi_livmessage
                        ('M03',date système ,v_ conumero ,v_manumero ,v_clnumero, user)
                sinon
                        insérer dans cdi_livmessage('M04',date système , N° CDE , user)
                fsi
        fsi
fin
```

3.7 Exo CDI : La procédure pro-arpo PLSQL_PROC_37

Soit ar_poicode, le code de poids d'un article

Objectif de la procédure : mettre à jour la colonne "poi_code" de la table "***cdi_article***" (créer cette colonne si besoin) pour les articles dont le poids est non vide. Exemple avec un extrait d'articles

ar_numero	ar_poids	poi_code
A10	120	
A11	100	1
A12		
A13	300	2
A14	300	
A15		
A16		
A17	600	3
A18	800	3
A19		
A20	50	
A21	60	1
A22	70	
A23		

Figure 3.7.1

Dans cet exemple,

- les articles rouges devraient avoir un poi_code. La procédure va effectuer cette opération.
- Les articles sont déjà traités
- les articles jaunes ne peuvent être traités

Voici la table "***cdi_poids***" :

POI_CODE	LIBELLE	POIDS_...	POIDS_MAX
1	PLUME	0	100
2	LEGER	101	500
3	MOYEN	501	2500
4	LOURD	2501	9999

Figure 3.7.2 . On suppose qu'aucun poids d'article ne dépasse 9999

Mettre à jour la table "***cdi_article***" en écrivant une procédure PL/SQL " ***PLSQL_PROC_37***". Le traitement prévu dans la procédure devra en plus de mettre à jour ***poi_code***, mettre à jour la table "***cdi_poimessage***". Cette table est caractérisée par les attributs suivants :

MESSAGE_124	TOTAL_LU	MESSAGE_3	LOGIN	DATE_JOU
table CDI ARTICLE traitée	23	partiellement	ERIC	11/04/18
table CDI_ARTICLE vide ou aucun traitement realisé	0	(null)	ERIC	16/03/17
table CDI_ARTICLE traitée	3	partiellement	ERIC	16/03/17
table CDI_ARTICLE vide ou aucun traitement realisé	0	(null)	ETU1_25	16/03/17
table cdi_poids vide	0	(null)	ERIC	15/03/17
table cdi_poids vide	0	(null)	ERIC	15/03/17
table CDI_ARTICLE traitée	20	en totalité	PATRICE	18/03/10

Figure 3.7.3

Message_124 ou message_3 peuvent prendre les valeurs suivantes

- message_124 { "table CDI_POIDS vide (1 a) | "table CDI_ARTICLE vide ou aucun traitement" (1b) | "table CDI ARTICLE traitée" (1c) }
  - nbre total de lignes lues
  - nbre de lignes traitées
- message_3 { "en totalité" (2a)| "partiellement" (2b) }

Le message 3 n'a d'intérêt qu'en cas (1c)

Le code pourra contenir plusieurs blocs (begin end;)

- Bloc1 :
  - On vérifie si la table ***cdi_poids*** est vide
  - Si oui on lève une exception et on écrit le message adéquat dans ***cdi_poimessage*** (1a)
  - Si non, on passe au bloc2
- Bloc2 :
  - On vérifie qu'il y a des articles de poids connu dont le " ***poi_code*** " est vide
  - Si non, on lève une exception et on écrit le message adéquat dans ***cdi_poimessage*** (1b)
  - Si oui, on passe au bloc3

- Bloc3 :
  - On copie les poids_max de la table cdi_poids dans une variable de type tableau vTabPoidsMax
  - On compte le nombre d'articles au total
  - Pour chaque article à mettre à jour, on compare le poids à vTabPoidsMax et on place le code du tableau dans article
  - On écrit le message adéquat dans ***cdi_poimessage*** (1c et 2a ou 2b)

Réaliser le code de cette procédure. Pour le bloc 3, l'utilisation d'un tableau est nécessaire (voir aide1 et aide 2).

Aide 1 : Utilisation d'un tableau

```
create or replace PROCEDURE ESSAI_TABLEAUX AS
TYPE tableau_numerique IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
tab1 tableau_numerique;

TYPE TYP_VAR_TAB is VARRAY(30) of number(4,1) ;
tab2 TYP_VAR_TAB := TYP_VAR_TAB(0,0,0,0,0);

BEGIN
  FOR i in 1..5 LOOP
   tab1(i) := i + (.2 * i ) ;
   tab2(i) := i + (.2 * i ) ;
 End loop ;
 FOR i IN tab1.FIRST..tab1.LAST LOOP
   dbms_output.put_line( 'tab1:'||tab1(i)||' tab2:'||tab2(i) ) ;
 End loop ;
END ESSAI_TABLEAUX;
/
```

Aide 2 : Déclaration des variables (début du code)

```
create or replace NONEDITIONABLE PROCEDURE PLSQL_PROC_37 AS
  vNbPoids number(2) := 0; -- utilisé dans le bloc1
  vNbArticlesLus     number(2):=0; -- pour compléter cid_poimessage et pour le bloc 2
  vMessage_1 cdi_poimessage.pom_message_124%type; -- message 1 pour cdi_poimessage
  vMessage_2 cdi_poimessage.pom_message_3%type := null; -- message 2 pour cdi_poimessage
```

```
  errPoidsVide exception; -- Exception bloc 1
  errPasDeTraitement exception; -- Exception bloc 2
  ------------------------------------------------------------------------
  cursor curCdiPoids is -- sélection des poids dans l'ordre croissant (bloc 3a)
    select poi_max from cdi_poids order by 1;
  vPoids_max cdi_poids.poi_max%type :=0;  -- poids max pour le curseur
  -- tpoids_max est un type de tableau de poids_max --------
  type TPOIDS_MAX is table of cdi_poids.poi_max%type index by binary_integer;
  vTabPoidsMax TPOIDS_MAX; -- tableau des poids_max de cdi_poids
  i  number(3) := 0; -- indice du tableau vTabPoidsMax
  ------------------------------------------------------------------------
  cursor curCdiArticle is -- sélection des poids non nul pour mise à jour du ar_poicode
    select ar_poids from cdi_article where  ar_poids is not null
    and poi_code is null for update of poi_code;
  vPoids cdi_article.ar_poids%type :=0;     -- utilisé avec curCdiArticle
  ------------------------------------------------------------------------
  vNbArticlesTotal number(2):=0; -- pour compléter cid_poimessage dans le bloc 3

BEGIN
    begin   -- BLOC 1 : recherche si cdi_poids est vide -
             dbms_output.put_line (chr(10) || 'BLOC 1');
             -- suite
     end ;  -- FIN BLOC 1
     -- suite
END ;
```

3.8 Exo CDI : Supplément : La procédure Recherche_Retards PLSQL_PROC_38

Objectifs de la procédure :

- Afficher les commandes pour lesquelles, il n'existe aucune livraison. Un retard est jugé excessif s'il dépasse 30 jours (afficher le nombre de jours de retard).
- Pour les livraisons existantes :
  - Afficher les lignes de livraison pour lesquelles, les quantités livrées sont différentes des quantités commandées

On pourra éventuellement utiliser les curseurs suivants :

```
create or replace PROCEDURE  PLSQL_PROC_38 AS
 cursor curPasLivraison is select co_numero,co_date from cdi_commande
 where co_numero not in ( select co_numero from cdi_livraison);

 cursor curIncomplet is select li_numero, ar_numero, lic_qtcmdee,lic_qtlivree
 from cdi_ligcde join cdi_livraison using(co_numero)
 join cdi_ligliv using(li_numero,ar_numero)
 where lic_qtcmdee<>lic_qtlivree;
```

4 Requêtes sur les tables

Les requêtes 4.1 et 4.2 sont fournies. Les tester et les commenter si nécessaire.

4.1 La projection

4.1.1 Afficher tous les clients
4.1.2 Afficher le prénom, nom pays et localité des clients
4.1.3 Rechercher la liste de toutes les localités de tous les clients
4.1.4 Même requête mais en supprimant les répétitions
4.1.5 Afficher les articles en limitant l'affichage à 10 articles.

4.2 La restriction (ou sélection)

4.2.1 Afficher tous les clients habitant "BORDEAUX"
4.2.2 Afficher tous les clients habitant dans une ville dont le nom commence par un "L"
4.2.3 Afficher tous les magasins dont le nom du gérant comprend un « A » ou un « I » en deuxième position

4.2.4 Afficher les fournisseurs dont le nom comprend un « N »
4.2.5 Sélectionner les articles dont le poids est supérieur à 500
4.2.6 Sélectionner les articles dont le poids est inférieur ou égal à 500
4.2.7 Sélectionner tous les articles pour lesquels le prix de vente est supérieur ou égal au double du prix d'achat
4.2.8 Sélectionner les articles de couleur rouge de poids supérieur à 100
4.2.9 Sélectionner les articles rouges et ceux de poids supérieurs à 100
4.2.10 On souhaite le contraire (complément) de la requête précédente. Pourquoi un nombre de résultats différent.
4.2.11 Sélectionner les articles étant soit, rouge et de poids supérieur à 100, soit de couleur verte
4.2.12 Afficher la liste des articles dont le prix d'achat est compris entre 10 et 25 euros
4.2.13 Afficher la liste des articles de couleur soit rouge soit verte
4.2.14 Afficher les articles dont la couleur n'est pas indiquée
4.2.15 Trier les articles selon l'ordre croissant de leur poids
4.2.16 Trier les clients dans l'ordre décroissant de leur prénom et à prénom égal, par nom croissant
4.2.17 Afficher le numéro, le nom et le prénom des clients, ainsi que le montant de leur chiffre d'affaires, si celui-ci dépasse 200

4.3 Fonctions et expressions

4.3.1 Afficher la marge bénéficiaire sur les produits dont le prix d'achat est supérieur à 10, par ordre de celle-ci
4.3.2 Pour tous les clients habitant à "PARIS", afficher le nom complet avec le nom, le prénom et la nationalité (concaténation)
Tester les requêtes agregation.sql
4.3.3 Calculer le délai moyen entre la commande et la date de livraison souhaitée, ainsi que le plus grand délai
4.3.4 Calculer le prix moyen (vente et achat) des articles (arrondir à trois chiffres après la virgule)
4.3.5 Afficher le prix d'achat de l'article le plus cher du stock (Option : afficher aussi le N° d'article)
4.3.6 Calculer le poids moyen, la marge maximum (la plus grande différence entre prix de vente et prix d'achat), la différence entre le prix de vente maximum et le prix d'achat maximum, le prix d'achat maximum, pour les articles dont la couleur n'est pas définie.

4.4 Requêtes sur les groupes

4.4.1 Compter le nombre de couleurs différentes.
4.4.2 Calculer le prix de vente moyen de chaque couleur d'articles.
4.4.3 Rechercher la couleur des articles dont le prix de vente moyen pour cette couleur est supérieur à 10.
4.4.4 Rechercher les magasins (ma_numero,ma_localite, ma_nom, ma_nom_gerant,ma_pernom_gerant), qui, pendant la période du 01/01/2007 au 01/01/2025 ont réalisé plus de 1 commande. Afficher également le nombre de commandes.
4.4.5 Afficher le nombre d'articles par couleur ainsi que le nombre d'articles au total.

	COUL...	NBPARCOULEUR	TOTAL
1	BLANC	8	24
2	BLEU	4	24
3	JAUNE	1	24
4	NOIR	2	24
5	ROUGE	4	24

Figure 4.4

4.4.6 Même question mais avec le total en bas.

4.5 La jointure et les sous-requêtes

4.5.1 Sélectionner les articles de couleur "ROUGE" et afficher le numéro, le poids et le nom du fournisseur
4.5.2 Afficher les clients qui ont commandé
4.5.3 Afficher les articles commandés par les clients qui habitent "CAEN"
4.5.4 Afficher les noms de tous les clients et leurs numéros de commandes pour ceux qui ont commandé.
4.5.5 Afficher la liste de tous les articles dont le prix d'achat est supérieur au prix d'achat de l'article "A07"
4.5.6 Afficher les commandes non prévues en livaison. Donner 3 versions (sous-requête, requête ensembliste, requête synchronisée). Utiliser Execute Explain Plan pour mesurer la performance.
4.5.7 Afficher les articles qui n'ont pas été commandés.
4.5.8 Rechercher tous les articles dont le poids est inférieur au poids de l'article "A02"
4.5.9 Rechercher les articles de même couleur que l'article "A14", et dont le poids est inférieur ou égal au poids moyen de tous les articles.
4.5.10 Donner la liste des fournisseurs qui proposent au moins un article de couleur "ROUGE"
- comment ferait-on avec la clause " au moins deux articles rouges" ?

4.5.11 onner le nom du magasin et le nom et prénom des gérants des magasins qui ont livré au moins un article "A02"
4.5.12 Donner la liste des articles dont le prix de vente est supérieur au prix de vente de l'article de couleur bleue le moins cher.
4.5.13 Projeter les clients ayant des commandes et des livraisons en utilisant 3 méthodes : jointure, sous-

requêtes, sous-requêtes synchronisées

4.6 Les opérateurs ensemblistes (Bonus)

4.6.1 Afficher par ordre croissant sur le numéro, les articles "ROUGE" et ceux commandés dans un magasin de "PARIS" (les valeurs "ROUGE" et "PARIS XX " seront affichés dans une colonne "couleur ville" en plus de ar_numero)

AR_NUMERO	couleur ville
A01	ROUGE
A02	PARIS 10E
A02	PARIS 5E

4.6.2 Afficher les articles "ROUGE" commandés dans un magasin de "PARIS"
4.6.3 Afficher les articles "ROUGE" sauf ceux commandés dans un magasin de "PARIS "

4.7 La mise à jour des données

(Ne pas faire de " commit " de façon à ne pas modifier durablement la base)

4.7.1 Insérer un article en remplissant toutes les colonnes. Le numéro d'article sera « A103 »
4.7.2 Insérer un article en ne remplissant qu'ar_numero, fo_numero, ar_nom. Le numéro d'article sera « A105 »
4.7.3 Insérer un article du fournisseur " ***G PAPIER*** ". La version complète permettra de générer le N° d'article (qui sera donc « A106 »).
4.7.4 Supprimer les articles rouges
4.7.5 Supprimer les articles du fournisseur " ***G PAPIER*** "
4.7.6 Mettre à jour la table cdi_ligcde de façon à ce que lic_pu des articles soit égale à ar_pv-10%
4.7.7 (Bonus) On suppose qu'une autre table possède des lignes de livraisons (CDI_LIGLIV_NEW) . L'objectif est de fusionner cette table avec CDI_LIGLIV. Pour atteindre cet objectif, on respectera les règles suivantes :

- Pour les lignes (LI_NUMERO,AR_NUMERO) de CDI_LIGLIV_NEW se trouvant déjà dans CDI_LIGLIV, on additionnera les quantités des deux lignes
- Pour les lignes (LI_NUMERO,AR_NUMERO) de CDI_LIGLIV_NEW ne se trouvant pas déjà dans CDI_LIGLIV, on insérera la ligne
- Pour réaliser cet exploit, il faut :
  - être en 2236
  - faire partie de la dream team de l'iut GON.
  - utiliser la commande MERGE INTO

Exemple :

```
drop table CDI_LIGLIV_NEW;
CREATE TABLE CDI_LIGLIV_NEW
(
 LIL_QTLIVREE NUMBER(3,0),
          LI_NUMERO CHAR(8),
          AR_NUMERO CHAR(8)
);
insert into CDI_LIGLIV_NEW values (40,'L9711   ','A10 ');
insert into CDI_LIGLIV_NEW values (3,'L9711   ','A07 ');
```

https://docs.oracle.com/cd/B28359_01/server.111/b28286/statements_9016.htm#SQLRF01606
https://oracle-base.com/articles/9i/merge-statement
https://www.oracletutorial.com/oracle-basics/oracle-merge

5 Les contraintes

5.1 Exercices de base

On imagine pour ces exercices de bases que l'on a recueilli dans une table **sondage** des informations sur des individus. On mémorise, un numéro (unique), la date de naissance de la personne sondée, la réponse à une première question, la réponse à une seconde question puis une valeur correspondant à la note que donne la personne sondée sur un élément donné.
Dans un premier temps, les données ont été enregistrées sans véritables règles (sans contraintes). Nous allons remédier à tout cela.

5.1.1 Construire une table sondage de cette façon :

```
create table sondage
(
 num number(3),
 date_naissance date,
```

num	date_naissance	reponse1	reponse2	val	compte_oracle
1	12/09/2000	oui	un peu	2	
2	13/06/2004	non	beaucoup	6	
3	01/04/1956	o	non	15	
25	16/02/1986	n	pas du tout	13	
26	06/02/1947	oui	à la folie	20	pirate14

```
  reponse1 char(20),
  reponse2 varchar2(20),
  val int,
  compte_oracle char(20) invisible
);
-- il y a mélange entre des données Oracle et des données Standards
```

5.1.2 Ajouter les lignes du tableau + 2 autres
5.1.3 Afficher toutes les lignes et colonnes de la table
5.1.4 Afficher la structure de la table
5.1.5 Exécuter `SET COLINVISIBLE ON;` puis afficher à nouveau la structure de la table
5.1.6 Convertir les types standards en type Oracle : reponse1 en chaîne de 3 caractères et transformer val en number(3))
5.1.7 Mettre à jour la table sondage avec les contraintes suivantes :

- le numéro d'observation est unique et non nul
- une date de naissance est comprise entre le 01/01/1850 et le 01/09/2018
- une première réponse ne peut être que 'oui' ou 'non'
- une deuxième réponse ne peut être que : 'pas du tout', 'un peu', 'beaucoup' ou 'ne se prononce pas'
- val est compris entre 1 et 10 inclus

5.1.8 Insérer quelques données dans cette table.
5.1.9 Créer une séquence pour l'auto-incrémentation du numéro d'observation
5.1.10 Insérer quelques données dans cette table.
5.1.11 Afficher la liste des séquences du dictionnaire des données
5.1.12 Montrer l'existence des contraintes dans le dictionnaire des données.
5.1.13 Supprimer les contraintes en utilisant une méta-requête.
5.1.14 Supprimer la table.
5.1.15 Exoplus : les clés composées. Tester et commenter chaque ligne

```
drop table tb;
drop table ta;
create table ta (a int, b int , constraint pk1 primary key (a,b));
create table tb (c int primary key, d int , e int);

alter table tb add constraint fk1 foreign key(d,e) references ta(a,b);
insert into ta values (1,1);
insert into ta values (1,3);
insert into ta values (2,3);
insert into ta values (2,3);
insert into ta values (5,null);
insert into tb values (1,1,1);
insert into tb values (3,3,5);
insert into tb values (4,null,null);
insert into tb values (5,1,null);
insert into tb values (6,null,8);
insert into tb values (7,2,3);
insert into tb values (8,2,3);
insert into tb values (9,4,3);
```

5.2 Exo CDI : Application sur l'intégrité référentielle

Exécuter cdi_etudiant_erreurs.sql

Les contraintes surlignées ne sont pas réalisables par des alter table. Elles seront traitées au chapitre 6

Comme il a été indiqué tout au long de l'étude, cette base est non seulement mal conçue au niveau structure mais aussi mal « alimentée ». On pourra ainsi trouver des incohérences (lignes manquantes, dupliquées) empêchant la création des clés primaires et étrangères. Saurez-vous les corriger ?

5.2.1 Ajouter des clés primaires à toutes les tables. Corriger les éventuelles anomalies.
5.2.2 Ajouter les clés étrangères en vérifiant la cohérence des données. Corriger les éventuelles anomalies. Créer également un ou deux index.
5.2.3 Ajouter des contraintes sur les colonnes en fonction des éléments suivants :

- "CDI_ Article" contient la liste des articles commercialisés.
  - Le stock et à 10 par défaut
  - Le stock doit être >= 0.
  - Tester avec ces lignes et commenter les résultats :

```
insert into cdi_article (ar_numero,fo_numero)  values ('A630','F01');
insert into cdi_article (ar_numero,fo_numero,ar_stock)  values ('A640','F01',null);
insert into cdi_article (ar_numero,fo_numero,ar_stock)  values ('A650','F01',-1);
select * from cdi_article order by ar_numero desc;
--rollback;
```

  - Le prix de vente doit être supérieur au prix d'achat.
  - Le poids d'un article ne peut dépasser 9998 g

- Les articles du fournisseur "F01" ne peuvent être de couleur "ROUGE"
- Tester avec cette dernière contrainte avec :

```
insert into cdi_article (ar_numero,fo_numero,ar_couleur) values ('A97','F01','ROUGE');
insert into cdi_article (ar_numero,fo_numero,ar_couleur) values ('A98','F02','ROUGE');
insert into cdi_article (ar_numero,fo_numero,ar_couleur) values ('A99','F01','BLEU');
rollback;
```

- Le prix de vente d'un article ne peut dépasser le double du prix moyen du même fournisseur.

- "**CDI Client**" contient la liste des clients potentiels.
  - La propriété "type" de "client" prend la valeur "PARTICULIER" en l'absence de valeur explicite.
  - Le chiffre d'affaires par défaut est 0.
  - Par contre, cet attribut ne pourra contenir que l'une des quatre valeurs mentionnées à savoir "PARTICULIER", "ADMINISTRATION", GRAND COMPTE" ou "PME".
- "**CDI Magasin**" contient la liste des magasins. Aucune contrainte demandée
- "**CDI_LigCde**", chaque occurrence contient les renseignements pour un article de commande.
  - La quantité livrée ne doit pas être supérieure à la quantité commandée.
  - La remise faite au client ne peut pas être supérieure à 20 % du prix catalogue.
  - La remise est au minimum de 10 % si la quantité commandée est supérieure ou égale à 100.
- "**CDI_LigLiv**", chaque occurrence contient les renseignements pour un article de la livraison. Il faut noter que le modèle contient une redondance : la quantité livrée est inscrite dans "CDI_LCDE". Elle a pour but de simplifier la gestion des commandes en cours. La mise à jour simultanée des deux attributs pourra être réalisée automatiquement par un déclencheur.

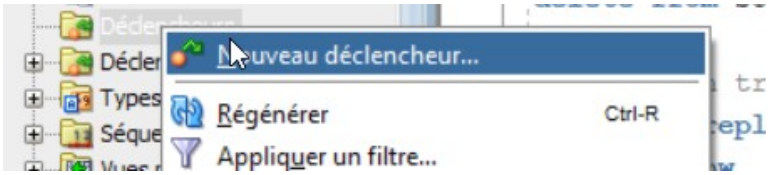
5.2.4 [Option fin TD-30 minutes] Tester les insertions dans des vues et justifier les éventuels messages d'erreurs

```
drop view v_article_fournisseur1;
create view v_article_fournisseur1  as
select fo.fo_nom, ar.fo_numero,ar_numero,ar_nom,ar_poids,ar_couleur,ar_stock,ar_pa,ar_pv
from cdi_article ar
join cdi_fournisseur fo on ar.fo_numero = fo.fo_numero;

drop view v_article_fournisseur2;
create view v_article_fournisseur2  as
select fo.fo_nom, fo.fo_numero,ar_numero,ar_nom,ar_poids,ar_couleur,ar_stock,ar_pa,ar_pv
from cdi_article ar
join cdi_fournisseur fo on ar.fo_numero = fo.fo_numero;

insert into v_article_fournisseur1 (fo_numero,ar_numero,ar_nom,ar_poids,ar_couleur,ar_stock,ar_pa,ar_pv)
values ('F06','A85','GOMME','25','BLANC',20,1,2);

insert into v_article_fournisseur2 (fo_numero,ar_numero,ar_nom,ar_poids,ar_couleur,ar_stock,ar_pa,ar_pv)
values ('F06','A86','GOMME','25','BLANC',20,1,2);

select * from user_updatable_columns where lower(table_name) like 'v_article_fournisseur%'
and column_name like '%NUMERO%';
```

5.2.5 Supprimer toutes les contraintes

6 Les déclencheurs (triggers)

Exécuter cdi_etudiant_correct.sql



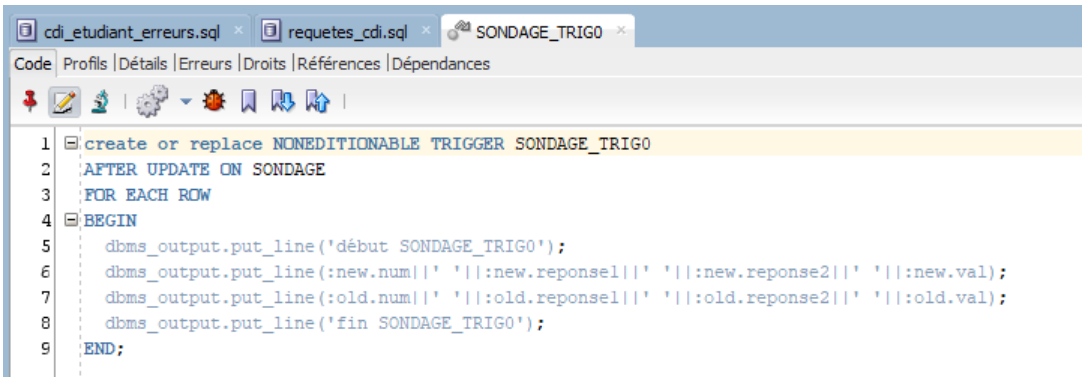
Rappels avec ce trigger (SONDAGE_TRIG0) hors sujet

2 modes :

- mode page sql

quand j'exécute avec le triangle vert, je compile (le / est important dans ce mode)

- mode edi



l'engrenage permet de compiler. On voit mieux les erreurs

6.1 Exercices de base

6.1.1 Créer la table **sondage** (supprimer l'ancienne si nécessaire) comme dans l'exercice 5-1-1 et en en faire une copie **sondage_copie**.

6.1.2 Réaliser un déclencheur nommé **SONDAGE_TRIG_612** qui se déclenche après insertion d'une donnée dans sondage. Ce déclencheur copie l'enregistrement dans **sondage_copie**. Compiler et tester le déclencheur.

- Test du trigger

```
insert into sondage values (74,to_date('01/09/2020','dd/mm/yyyy'),'oui','un peu',9);
insert into sondage values (70,to_date('01/09/2013','dd/mm/yyyy'),'oui','non',9);
select * from sondage_copie;
```

6.1.3 Compléter ce déclencheur renommé **SONDAGE_TRIG_613** (la version trigger ligne) pour qu'il se déclenche après mise à jour d'une donnée dans sondage. Dans la version complète, on mettra à jour la table sondage_copie sur mise à jour de sondage.

6.1.4 Réaliser un déclencheur nommé **SONDAGE_TRIG_614** qui se déclenche avant insertion ou mise à jour d'une donnée dans sondage. Ce déclencheur provoquera une exception appropriée en cas de tentative d'insertion d'un numéro vide, d'une date non comprise entre le 01/01/1850 et aujourd'hui, d'une valeur non comprise entre 0 et 10.

- Test du trigger

```
insert into sondage values (9    ,to_date('20/12/2009','dd/mm/yyyy'),'oui','non',9);
insert into sondage values (10   ,to_date('20/12/2019','dd/mm/yyyy'),'oui','non',14);
insert into sondage values (11   ,to_date('20/12/2021','dd/mm/yyyy'),'oui','non',9);
insert into sondage values (null ,to_date('20/12/2019','dd/mm/yyyy'),'oui','non',9);
```


petit coup de pouce :

```
create or replace trigger "SONDAGE_TRIG_614"
BEFORE
insert or update on "SONDAGE"
for each row
declare
 i int;
 erreur_num EXCEPTION;
 erreur_date EXCEPTION;
 erreur_val EXCEPTION;
begin
 i := 0;
 if :new.num is null then
```

```
  raise erreur_num;
end if;
...
```

6.1.5 Créer une table bilan permettant de compter le nombre de chaque réponse1 et réponse2. Insérer une première ligne avec des 0 dans les compteurs.

```
create table bilan
(
rep1_oui number(4),rep1_non number(4),rep2_pdtt number(4),rep2_unpe number(4),
rep2_bcps number(4),rep2_nspp number(4)
);
```

Réaliser un déclencheur nommé **SONDAGE_TRIG_615** qui se déclenche après insertion ou mise à jour d'une donnée dans sondage. Ce déclencheur mettra à jour le nombre de réponses dans la table bilan. *Pour gagner du temps, ne travailler que sur reponse1.*

```
-- pour tester le déclencheur
insert into sondage values (10,to_date('20/12/2009','dd/mm/yyyy'),'oui','un peu',9);
insert into sondage values (11,to_date('20/12/2009','dd/mm/yyyy'),'oui','pas du tout',6);
insert into sondage values (12,to_date('20/12/2009','dd/mm/yyyy'),'oui','non',4);
insert into sondage values (13,to_date('20/12/2009','dd/mm/yyyy'),'non','non',9);
select * from bilan;
delete from bilan;
select * from sondage;
delete from sondage;
update sondage set reponse1='non' where val=4;
update sondage set reponse1='oui' where val=8;
```

6.2 Exo cdi : Réalisation des contraintes manquantes

Certaines contraintes n'ont pu être ajoutées statiquement car Oracle 21C ne supporte pas les sous-requêtes dans la condition check. Le but est de gérer ces contraintes par des déclencheurs qui vont se déclencher sur insertion ou mise à jour des tables concernées.

- ✗ Réaliser un déclencheur pour la mise à jour des articles : **CDI_ARTICLE_TRIG_MAJ** pour tenir compte de la contrainte manquante. Ne s'intéresser qu'aux insertions et ne pas tenir compte du cas " table article vide ".
  - 6_2A) Le prix de vente d'un article ne peut dépasser le double du prix de vente moyen du même fournisseur.

```
insert into cdi_article values('A80','F01','','Calculatrice',50,'BLEU',5,25,20);
insert into cdi_article values('A81','F01','','Calculatrice',50,'BLEU',5,25,900);
update cdi_article set ar_pv=900 where ar_numero='A80'; -- pas avec trigger ligne
delete from cdi_article where ar_numero = 'A80';
delete from cdi_article where ar_numero = 'A81';
```

- ✗ Réaliser un déclencheur pour la mise à jour ligcde : **CDI_LIGCDE_TRIG_INSERT** pour tenir compte de la contrainte manquante. Ne s'intéresser qu'aux insertions.
  - 6_2_B) La remise faite au client ne peut pas être supérieure à 20 % du prix catalogue (ar_pv).
  - 6_2_B) La remise est au minimum de 10 % si la quantité commandée est supérieure ou égale à 100.

```
delete from cdi_ligcde where co_numero='C0901';
delete from cdi_commande where co_numero = 'C0901';
select * from cdi_commande where co_numero  = 'C0901';
insert into cdi_commande (co_numero, co_date, cl_numero, ma_numero) values ('C0901',sysdate,'C01','M02');
select * from cdi_article where AR_NUMERO = 'A02'; -- 29 €
select * from cdi_article where AR_NUMERO = 'A01'; -- 10 €

insert into cdi_ligcde (lic_QTCMDEE,LIC_QTLIVREE,LIC_PU,co_numero,ar_numero,date_liv)
values (1,1,10,'C0901','A02',sysdate); -- trop forte
insert into cdi_ligcde (lic_QTCMDEE,LIC_QTLIVREE,LIC_PU,co_numero,ar_numero,date_liv)
values (1,1,40,'C0901','A02',sysdate); -- OK
insert into cdi_ligcde (lic_QTCMDEE,LIC_QTLIVREE,LIC_PU,co_numero,ar_numero,date_liv)
values (111,1,37,'C0901','A02',sysdate); -- trop faible
insert into cdi_ligcde (lic_QTCMDEE,LIC_QTLIVREE,LIC_PU,co_numero,ar_numero,date_liv)
values (56,1,9,'C0901','A01',sysdate); -- OK
```

```
select * from cdi_ligcde where co_numero  = 'C0901';
```

6.3 Exo cdi : Mise à jour automatique des lignes de commandes

✗(6.3a : **CDI_LIVRAISON_TRIG_INS**) A partir d'un nouvel enregistrement dans cdi_livraison, créer **les** enregistrements dans cdi_ligliv (en tenant compte de cdi_ligcde). Par défaut, les quantités livrées seront placées à 0.

✔ utiliser un curseur pour lire cdi_ligcde

✔ on pourra vérifier le déclencheur (quand il sera opérationnel) avec le jeu de test suivant :

```
delete from cdi_commande where co_numero='C9999';
delete from cdi_ligcde where co_numero='C9999';
delete from cdi_livraison where co_numero='C9999';
delete from cdi_ligliv where li_numero='L9901';

insert into cdi_commande (co_numero,ma_numero,cl_numero,co_date) values('C9999','M99','C99','13/11/2009');
insert into cdi_ligcde (ar_numero,co_numero,date_liv,lic_qtcmdee,lic_qtlivree,lic_pu) values
(
 'A07',
 'C9999',
 null,
 5,
 0,
 (select ar_pv from cdi_article where ar_numero='A07')
);
-- faire plusieurs lignes de commande (ligcde)

insert into cdi_livraison (li_numero,ma_numero,co_numero,cl_numero,date_liv) values
(
  'L9901',
  (select ma_numero from cdi_commande where co_numero='C9999'),
  'C9999',
  (select cl_numero from cdi_commande where co_numero='C9999'),
  '18/11/2009'
);




select * from cdi_commande order by co_numero desc ;
select * from cdi_ligcde order by co_numero desc;
select * from cdi_article order by ar_numero;
select * from cdi_livraison order by li_numero desc;
select * from cdi_ligliv order by li_numero desc;
```

✗ (6.3b : CDI_LIGLIV_TRIG_MAJ) A partir d''une mise à jour dans cdi_ligliv, concevoir un déclencheur capable de mettre automatiquement à jour

✔ la date de livraison dans cdi_livraison

✔ la date de livraison dans cdi_ligcde.

✔ la quantité livrée dans cdi_ligcde.

✗ [Obligation] Ce déclencheur ne se déclenche que sur mise à jour de la quantité livrée.

✗[Obligation] Ce déclencheur ne se déclenche que si la nouvelle quantité livrée est différente de l'ancienne.

✗[Option] Compléter ce déclencheur pour mettre à jour le stock dans cdi_article. Un stock est à <null> si aucun stock n'est géré pour un article.

✗[Option] Générer une exception si le stock n'est pas suffisant..

✗[Option] Si la nouvelle quantité livrée est inférieure à l'ancienne, c'est du vol !

6.4 Exo cdi : Mise à jour des fournisseurs: cdi_fournisseur_trig_maj

On souhaite effectuer des sauvegardes des données Fournisseur. A chaque fois qu'un fournisseur est mis à jour (suppression ou modification), un enregistrement dans une table cdi_fournisseur_maj qui contient le fonumero, le fonom, la date de modification, le type de modification, le nom de l'utilisateur qui a effectué cette modification.

Réaliser le code de ce déclencheur **CDI_FOURNISSEUR_TRIG_MAJ.**

6.5 Déclencheur ldd : trigger_journal

En utilisant le tableau 11.5 de la documentation " database_triggers.PDF ", réaliser un déclencheur ldd (trigger_journal) se déclenchant sur des opérations create, drop ou alter et qui met un jour une table journal avec les informations suivantes :

JOURNAL

- operation varchar2(20), type_objet varchar2(20)
- nom_objet varchar2(20), quand date, qui varchar2(32)

Tester le déclencheur avec des exemples classiques :

```
drop table td2_access;
create table td_oracle( i int);
create table td0_access( i int);
create table tmp( i int);
create view  vt_temp as select * from tmp;
create or replace synonym tmp2 for tmp;
alter table tmp add b int;
alter table tmp add c int;
drop table tmp;
```

6.6 Déclencheur d'instance

Mettre au point un système permettant d'enregistrer dans une table journal_connexion la durée où la session a été ouverte. Pour simplifier on suppose qu'au maximum, une seule session est ouverte pour votre schéma.

On peut utiliser une table journal comme celle-ci

```
drop table journal_connexion;
create table journal_connexion
(connexion date, deconnexion date, duree number(10,1), utilisateur varchar2(20));
desc journal_connexion;
```