

Département Génie Informatique

BD50

TP5 : Développement PL/SQL
Avec Oracle SQL Developer

Gestion Commerciale

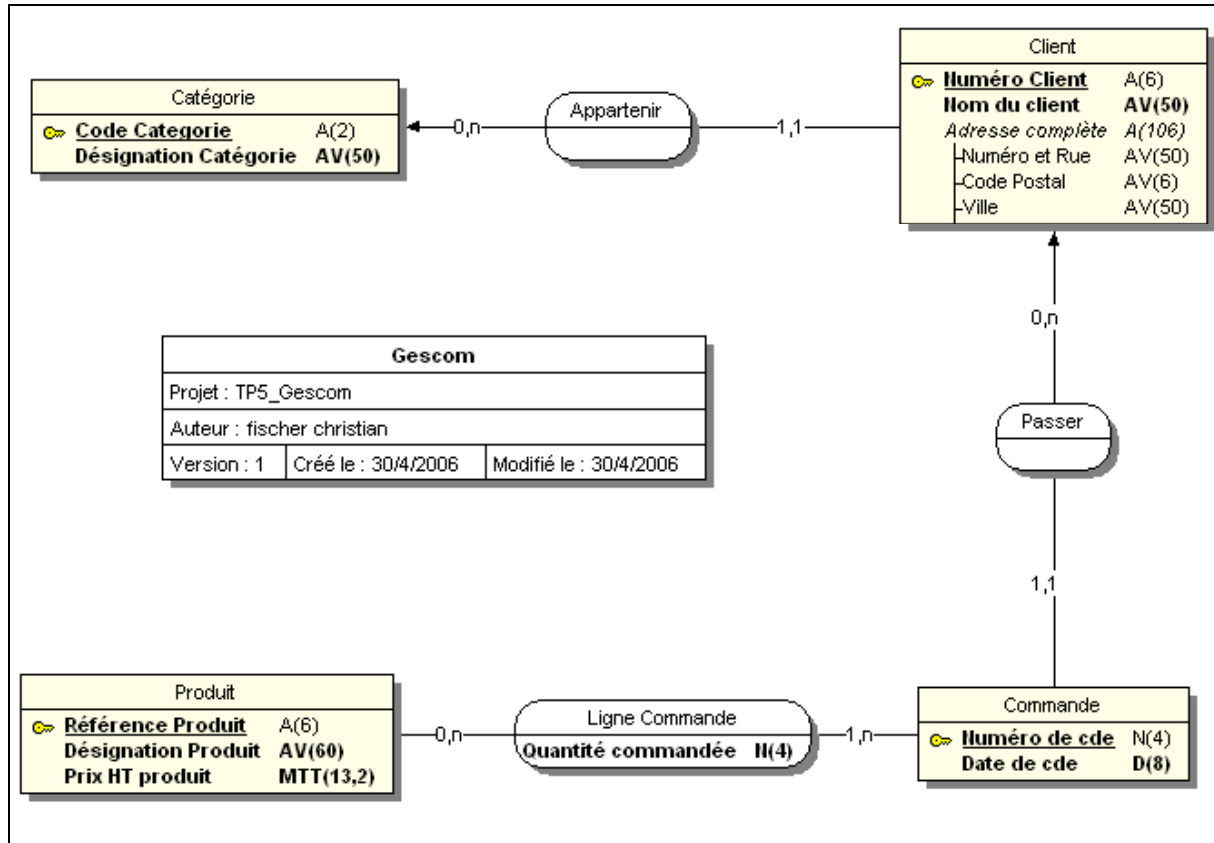
Françoise HOUBERDON & Christian FISCHER

Copyright © Avril 2007

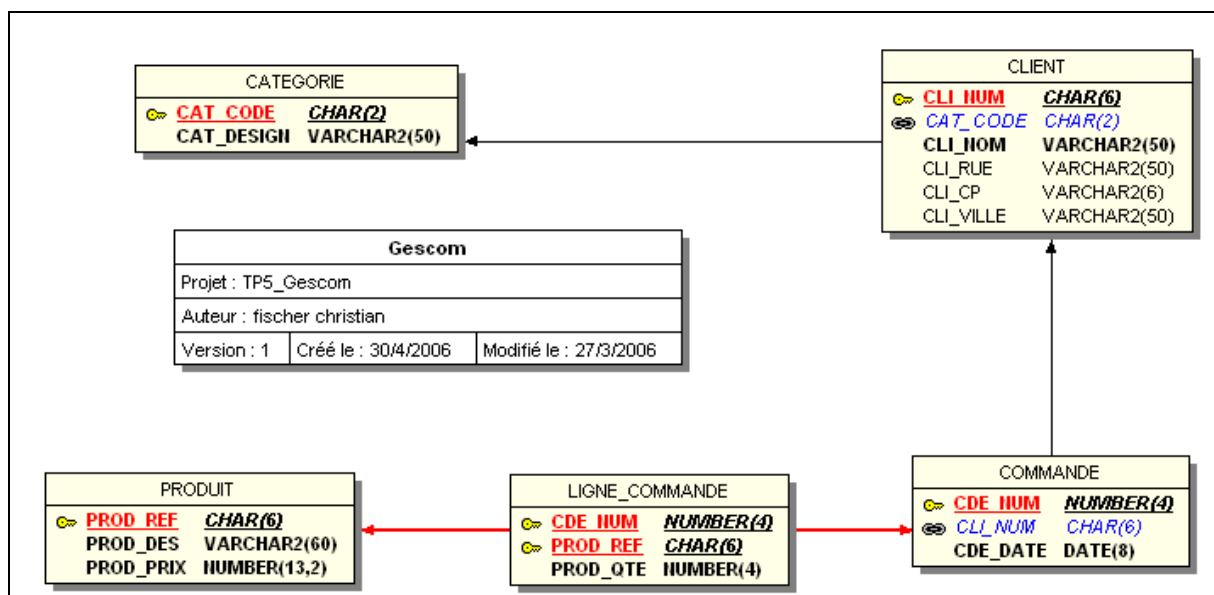
Présentation de la gestion commerciale

Après avoir modélisé sa gestion commerciale dont le modèle conceptuel de données, utilisant le formalisme entité-association, est présenté ci-dessous :

Modèle entité-association



Modèle logique de données relationnel normalisé associé est le suivant :



(Modification de la mise en forme graphique par défaut dans WinDesign)

Travail à faire :

Étape 1 : Utilisation de WinDesign pour générer le script SQL de création des tables et index l'application

En utilisant WinDesign, vous devez recréer le modèle entité-association, puis générer le modèle relationnel associé.

Étape 2 : Création du compte applicatif Oracle

La base de données est implantée sur un serveur Oracle.

Le nom du compte de connexion à créer est : **votrenom_gescom**

Le mot de passe du compte est : **votrenom**

Le tablespace par défaut pour le stockage des tables est : BD50_DATA

Le tablespace temporaire est : TEMP

Les privilèges à accorder sont : Connect et Resource

Ancien serveur Oracle

La chaîne de connexion est : **GI_PROD19.UTBM.FR**

L'url de iSQL*Plus est : **http://pc-gi-19:5560/isqlplus**

Nouveau serveur Oracle (installé le 27/3/07)

La chaîne de connexion est : **GI_PROD203.UTBM.FR**

L'url de iSQL*Plus est : **http://pc-gi-203:5560/isqlplus**

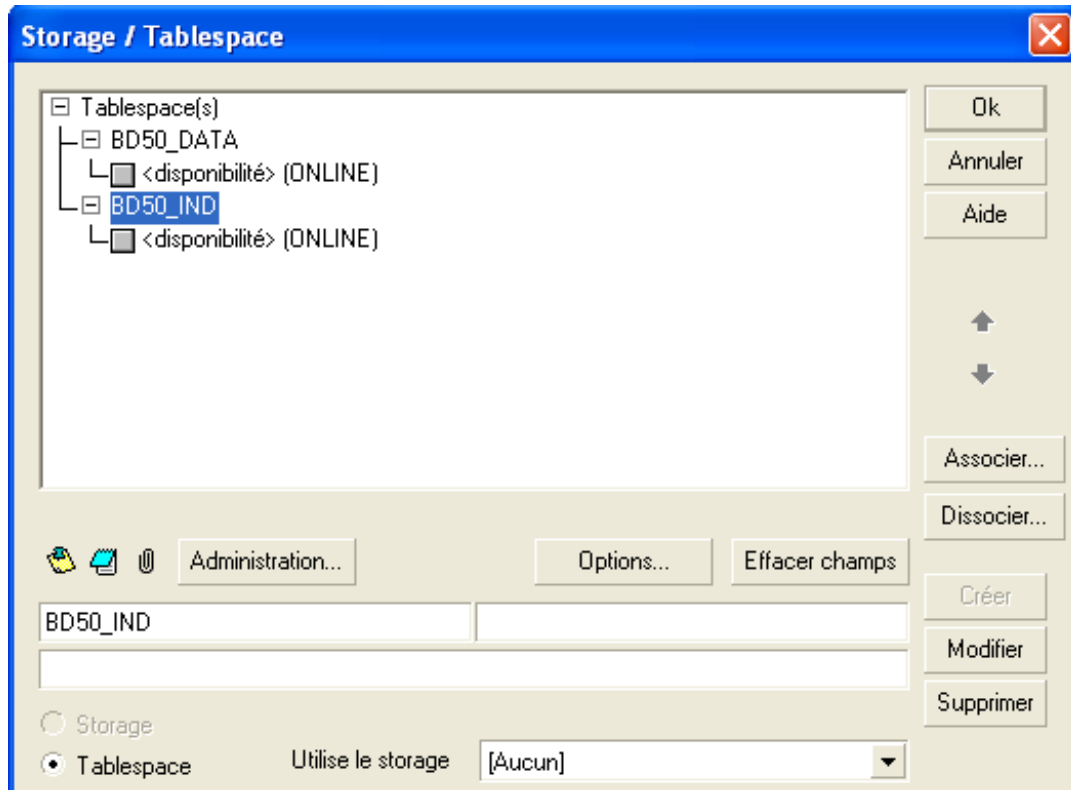
Le tablespace par défaut pour le stockage des index est : BD50_IND

Le tablespace d'index doit être intégré dans le script de création. Pour les clés primaires et les index vous devez utiliser cet index.

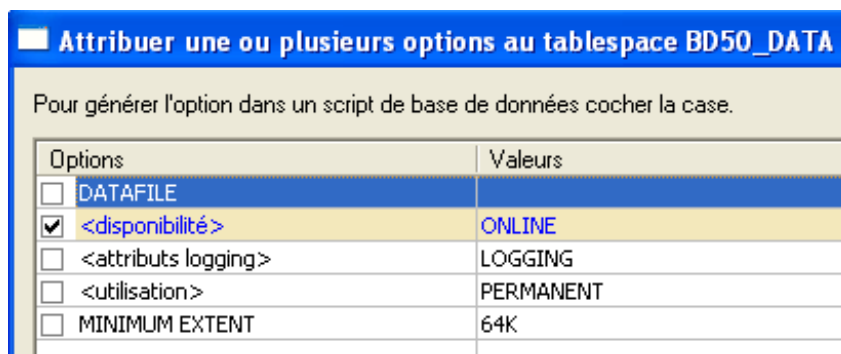
Étape 3 : Génération du script SQL

Dans le modèle relationnel, utilisant le SGBD Access, créez les tablespaces nécessaires à la génération du script SQL.

Menu Base de données/Tablespace-Storage



Pour chaque tablespace, définissez l'option ONLINE, à partir du bouton Options de l'écran ci-dessus.



Association du tablespace d'index aux clés primaires et index.

Pour chaque table accéder aux propriétés de la table. Sélectionner les différents index, puis pour chaque index cliquer sur bouton Storage/Tablespace

Détails de la table : CLIENT

[Aucun] Afficher informations

| Nom logique | | | | | | | | | | Nom co |
|-------------------------------|---|---|--|--|---|--|---|---|--|--------|
| ● CLI_NOM | ✓ | | | | | | | ✓ | | Nom |
| ● CLI_RUE | ✓ | | | | | | | | | Numé |
| ● CLI_CP | ✓ | | | | | | | | | Code |
| ● CLI_VILLE | ✓ | | | | | | | | | Ville |
| Clés étrangères (1) | | | | | | | | | | |
| + ● FK_CLIENT_CATEGORIE (...) | | | | | | | | ✓ | | |
| Index (2) | | | | | | | | | | |
| + ● I_PK_CLIENT | ✓ | ✓ | | | | | | | | |
| ↑ CLI_NUM | ✓ | ✓ | | | | | ✓ | ✓ | | Num |
| + ● I_FK_CLIENT_CATEGORIE | ✓ | | | | ✓ | | | | | |

Définition Car. étendues Administration

Composition... Effacer champs

I_PK_CLIENT

Stéréotype [aucun]

Unique Options :

Type d'index... Index sur clé primaire

Affichage : Respecter les options

Storage / Tablespace...

Générer le script SQL

Dans les options de génération, conservez les options suivantes

Génération d'un script de base de données

Général Options

Base de données

- Création
- Suppression
- Ouverture
- Fermeture

Contraintes

- Clé primaire
- Clé alternative
- Clé étrangère
- Contraintes de valeurs
- Déclaratives

Autres options

- Majuscules
- Minuscules
- ANSI
- Plusieurs fichiers
- Création règles

Table

- Création
- Suppression
- Commentaire table
- Commentaire attributs
- même si libellé vide

Index

- Sur clé primaire
- Sur clé alternative
- Sur clé étrangère
- Autres index
- Suppression

TableSpace / Storage

- Création TableSpace
- Suppression TableSpace
- Création Storage
- Suppression Storage
- Affectation aux objets

Vue SQL / Requête

- Création vue SQL
- Suppression

Procédure / Fonction

- Création
- Suppression

Type Utilisateur

- Création
- Suppression
- Création des séquences

Vérification de la génération des tablespaces sur les index

Exemple pour la table : CLIENT

```
CREATE TABLE CLIENT
(
  CLI_NUM CHAR(6) NOT NULL,
  CAT_CODE CHAR(2) NOT NULL,
  CLI_NOM VARCHAR2(50) NOT NULL,
  CLI_RUE VARCHAR2(50) NULL,
  CLI_CP VARCHAR2(6) NULL,
  CLI_VILLE VARCHAR2(50) NULL
,
  CONSTRAINT PK_CLIENT PRIMARY KEY (CLI_NUM)
  USING INDEX TABLESPACE BD50_IND
) ;

-----
--          INDEX DE LA TABLE CLIENT
-----

CREATE INDEX I_FK_CLIENT_CATEGORIE
ON CLIENT (CAT_CODE ASC)
TABLESPACE BD50_IND ;
```

Remarque : Vérifier la non génération du tablespace d'index sur les clés étrangères dans l'instruction Alter Table Si nécessaire supprimer la clause dans le script.

Étape 4 : Transfert des données

Exécution du script SQL de création des tables et index généré.

```
SQL> connect votrenom_gescom/votremdp@gi_prod203.utbm.fr
```

```
SQL> show user
```

```
SQL> @u:\..\crebase.sql
```

```
SQL> select * from tab ;
```

Transfert des données à partir du compte GESCOM dans vos tables en utilisant des instructions SQL

Vérification l'accès aux tables du compte GESCOM

```
SQL> connect votrenom_gescom/votremdp@gi_prod203.utbm.fr
```

```
SQL> show user
```

```
SQL> select table_name from all_tables where owner='GESCOM' ;
```

Vérifier l'ordre des colonnes dans votre table et la table du compte Gescom.
En cas de différence, modifiez le code SQL ci-dessous.

Remplacer nomtable par le nom des tables dans le schéma

```
SQL> Insert into nomtable select * from gescom.nomtable ;
```

Suivi de la validation de transaction

```
SQL> Commit;
```

Extrait du jeu d'essai disponible dans le compte GESCOM

PO_GCOM CATEGORIE

Columns Data Indexes Constraints Grants Statistics Column Statistics

Sort... Filter:

| CAT_CODE | CAT_DESIGN |
|----------|----------------|
| 1 GR | GROSSISTE |
| 2 GS | GRANDE SURFACE |
| 3 DE | DETAILLANT |

PO_GCOM CLIENT

Columns Data Indexes Constraints Grants Statistics Column Statistics Triggers Dependencies Details SQL

Sort... Filter:

| CLI_NUM | CAT_CODE | CLI_NOM | CLI_RUE | CLI_CP | CLI_VILLE |
|-----------|----------|-------------------------|------------------------------|--------|-------------------|
| 1 411001 | GR | ABI CARAVANES | 35 rue de Condorcet | 95100 | ARGENTEUIL |
| 2 411002 | DE | KIT LOISIRS | 135 rue Jean-Baptiste Potin | 92170 | VANVES |
| 3 411003 | DE | LEVOYAGEUR | 36 boulevard Gambetta | 75020 | PARIS |
| 4 411004 | DE | SOCANOR | 66 avenue du Général Leclerc | 93100 | MONTREUIL |
| 5 411005 | GR | NOTIN | 120 avenue Henri Barbusse | 93260 | LES LILAS |
| 6 411006 | GR | ESSONNE CARAVANES | 70 rue de Fleury | 91310 | LONGPONT SUR ORGE |
| 7 411007 | DE | LABALETTE | 140 avenue Victor Hugo | 92380 | GARCHES |
| 8 411008 | DE | CENTRE EUROPE CARAVANES | 23 rue Galvani | 91300 | MASSY |
| 9 411009 | GS | VVL | 49 avenue de l'Europe | 91100 | CORBEL |
| 10 411010 | GS | PSL | 67 rue d'Etienne d'Orves | 91170 | VIRY CHATILLON |
| 11 411011 | GS | SA MAP-HOME SAMCO | 65 rue de Mesly | 94000 | CRETEIL |
| 12 411012 | DE | RELAIS LOISIRS | 80 rue du Boucher | 94150 | RUNCIS |
| 13 411013 | DE | MOBILVETTA | 120 avenue Daumesnil | 75012 | PARIS |
| 14 411014 | DE | Client 411014 | 12 quai de la gare | 91000 | MASSY |

PO_GCOM COMMANDE

Columns Data Indexes Constraints Grants Statistics Column Statistics

Sort... Filter:

| CDE_NUM | CLI_NUM | CDE_DATE |
|---------|-----------|----------|
| 1 | 1 411001 | 05/03/97 |
| 2 | 2 411001 | 19/05/97 |
| 3 | 3 411002 | 22/12/97 |
| 4 | 4 411002 | 09/02/97 |
| 5 | 5 411003 | 15/02/97 |
| 6 | 6 411003 | 17/02/97 |
| 7 | 7 411003 | 31/05/97 |
| 8 | 8 411003 | 13/08/97 |
| 9 | 9 411003 | 03/08/97 |
| 10 | 10 411004 | 19/03/97 |

| PO_GCOM | | PRODUIT | |
|---------|----------|-------------------------------|-------------|
| Columns | Data | Indexes | Constraints |
| Sort... | | Filter: | |
| | PROD_REF | PROD_DES | PROD_PRIX |
| 1 | M00645 | Désignation du produit M00645 | 600 |
| 2 | M00646 | Désignation du produit M00646 | 700 |
| 3 | M00647 | Désignation du produit M00647 | 800 |
| 4 | M00648 | Désignation du produit M00648 | 900 |
| 5 | M00649 | Désignation du produit M00649 | 1000 |
| 6 | M00650 | Désignation du produit M00650 | 100 |
| 7 | N00651 | Désignation du produit N00651 | 200 |
| 8 | N00652 | Désignation du produit N00652 | 300 |
| 9 | N00653 | Désignation du produit N00653 | 400 |
| 10 | N00654 | Désignation du produit N00654 | 500 |

| PO_GCOM | | LIGNE_COMMANDE | |
|---------|---------|----------------|-------------|
| Columns | Data | Indexes | Constraints |
| Sort... | | Filter: | |
| | CDE_NUM | PROD_REF | PROD_QTE |
| 1 | 1 | J00465 | 2 |
| 2 | 2 | E00212 | 10 |
| 3 | 2 | M00646 | 6 |
| 4 | 2 | O00713 | 7 |
| 5 | 2 | R00882 | 8 |
| 6 | 3 | J00484 | 9 |
| 7 | 3 | K00524 | 7 |
| 8 | 3 | K00548 | 5 |
| 9 | 3 | P00753 | 5 |
| 10 | 3 | P00766 | 7 |

Étape 3 : Développement des procédures et fonctions stockées

| | |
|---|--|
| A | <p>Etude et création d'une procédure stockée à l'aide du bloc note et de SQL*PLUS ou de ISQL*PLUS</p> <p>La table des produits se compose d'une référence de produit (PROD_REF), d'une désignation (PROD_DES) et d'un prix unitaire de vente (PROD_PRIX).</p> <p>La procédure AugmenterPrix se présente ainsi :</p> |
| <pre>CREATE OR REPLACE PROCEDURE AugmenterPrix(vprodref CHAR) IS px NUMBER(6,2); BEGIN SELECT PROD_PRIX INTO px FROM PRODUIT WHERE PROD_REF = vprodref; IF px > 500.00 THEN px := px * 1.1; ELSE px := px * 1.2 ; END IF; UPDATE PRODUIT SET PROD_PRIX = px WHERE PROD_REF = vprodref; COMMIT; END AugmentePrix; /</pre> | |
| Quel est le rôle de cette procédure ? | |
| Quel est le rôle des identificateurs suivants ? | |
| Vprodref | |
| px | |
| Fournir les instructions d'affectation ? | |
| Quelles sont les structures de cette procédure ? <input type="checkbox"/> Itérative <input type="checkbox"/> Conditionnelle <input type="checkbox"/> Séquence | |
| Quel est le rôle de l'instruction COMMIT ? Cette instruction est-elle obligatoire dans cette procédure ? | |
| Créer et tester la procédure. Insérer le produit suivant dans la table des produits : ('TV70CO', 'Télévision couleur 70 cm ', 1000) | |
| Fournir l'instruction qui permet d'exécuter cette procédure pour le produit numéro TV70CO ? | |

B**Compléter le code PL/SQL d'une procédure stockée paramétrée.**

Le nom de la procédure est StatProd. Elle permet de calculer la quantité totale commandée (qte) et le nombre de clients (nbcli) ayant commandé un produit.

Les paramètres de cette procédure sont :

Entrée : prodref de type char

Sortie : qte et nbcli de type numérique

```
CREATE OR REPLACE PROCEDURE _____
(
  _____
)
IS
BEGIN
  SELECT SUM(PROD_QTE) INTO _____
  FROM LIGNE_COMMANDE
  WHERE PROD_REF = prodref;

  SELECT COUNT(DISTINCT CLI_NUM) INTO _____
  FROM LIGNE_COMMANDE, COMMANDE
  WHERE LIGNE_COMMANDE.CDE_NUM = COMMANDE.CDE_NUM
  AND LIGNE_COMMANDE.PROD_REF = prodref;

END StatProd;
/
```

Compléter le code de la procédure stockée ci-dessus, puis créer et tester la procédure

Insérer le produit suivant dans la table des produits :
('TVCO84', 'Télévision couleur 84 cm ', 1300)

A partir de l'interface Sql*Plus pour Windows, fournir l'instruction qui permet d'exécuter cette procédure en utilisant les variables hôtes VQTE et VNBCLI et la référence de produit 'TVCO84' ?

-- Déclaration des variables hôtes

VARIABLE VQTE NUMBER

VARIABLE VNBCLI NUMBER

-- Exécution de la procédure (une variable hôte doit être préfixée par :)

-- Affichage des valeurs des variables

PRINT VQTE

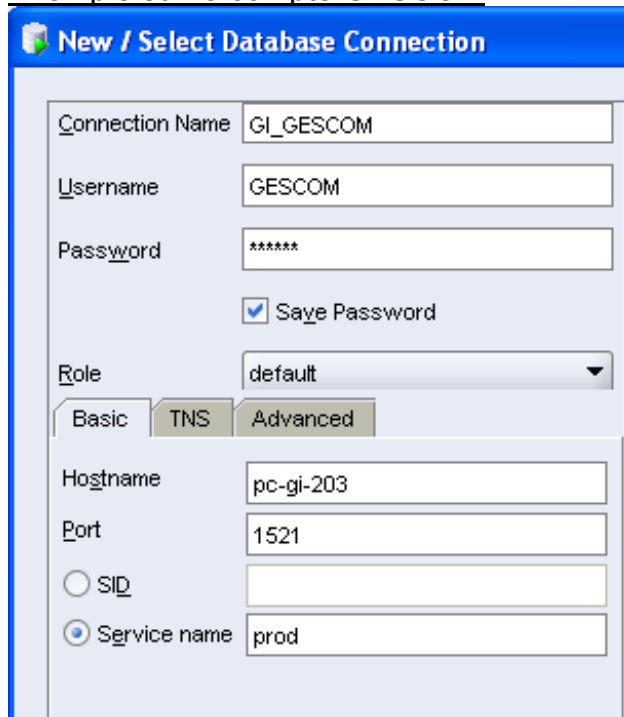
PRINT VNBCLI

C**Création d'une procédure stockée avec SQL Developer****Ajouter_produit : permettant de créer un nouveau produit**

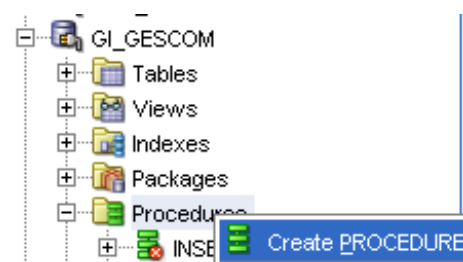
A partir de SQL Developer créer une nouvelle procédure stockée.

1. Création d'une nouvelle connexion de base de données
2. Fournir les informations de connexion **votrenom_gescom**, le mot de passe et la chaîne de connexion.

Exemple sur le compte GESCOM



Dans le navigateur sélectionner **Procédures** puis dans le menu contextuel sélectionner **Create Procedure**.



Le nom de la procédure est : **ajouter_produit**

Dans l'éditeur d'unité de programme compléter le code PL/SQL à partir de l'exemple ci-dessous :

```
CREATE OR REPLACE PROCEDURE GESCOM.AJOUTER_PRODUIT
(v_prodfref produit.prod_ref%TYPE
,v_proddes produit.prod_des%TYPE
,v_prodprix produit.prod_prix%TYPE)
AS
BEGIN
INSERT INTO produit (PROD_REF, PROD_DES, PROD_PRIX)
VALUES      (v_prodfref, v_proddes, v_prodprix);
COMMIT;
END;
```

Compiler puis enregistrer la procédure.

Exécuter la procédure à l'aide du bloc PL/SQL

Run PL/SQL

| Parameter | Data Type | Mode |
|------------|--------------|------|
| V_PRODREF | CHAR(6) | IN |
| V_PRODDDES | VARCHAR2(60) | IN |
| V_PRODPRIX | NUMBER | IN |

```
DECLARE
V_PRODREF CHAR(6);
V_PRODDDES VARCHAR2(60);
V_PRODPRIX NUMBER;
BEGIN
V_PRODREF := 'TYC084';
V_PRODDDES := 'Télévision couleur 84 cm';
V_PRODPRIX := '990';

AJOUTER_PRODUIT(
V_PRODREF => V_PRODREF,
V_PRODDDES => V_PRODDDES,
V_PRODPRIX => V_PRODPRIX
);
END;
```

Vérifier l'exécution de votre procédure en consultant la table produit :
select * from produit ;

Exécuter de nouveau la procédure avec les mêmes valeurs. Que constatez-vous ?

D**Création d'une procédure stockée****Maj_produit : permettant de modifier la description d'un produit**

A partir de SQL Developer, créer une nouvelle procédure stockée.
Dans le navigateur sélectionner : **Procédures** puis dans le menu contextuel sélectionner **Create Procedure**.

Le nom de la procédure est : maj_produit

Dans l'éditeur compléter le code PL/SQL à partir de l'exemple ci-dessous :

```
CREATE OR REPLACE PROCEDURE maj_produit
(v_prodref IN produit.prod_ref%TYPE,
 v_proddes IN produit.prod_des%TYPE)
IS
BEGIN
UPDATE produit
  SET  prod_des = v_proddes
 WHERE prod_ref = v_prodref;
IF SQL%NOTFOUND THEN
  RAISE_APPLICATION_ERROR(-20202,'Pas de produit mis à jour. ');
ELSE
  COMMIT;
END IF ;
END maj_produit;
```

Compiler, puis enregistrer la procédure.

Exécuter la procédure avec les valeurs suivantes :

V_PROD_REF='TVC84'

V_PROD_DES='Télévision Couleur 84 cm 4/3'

E

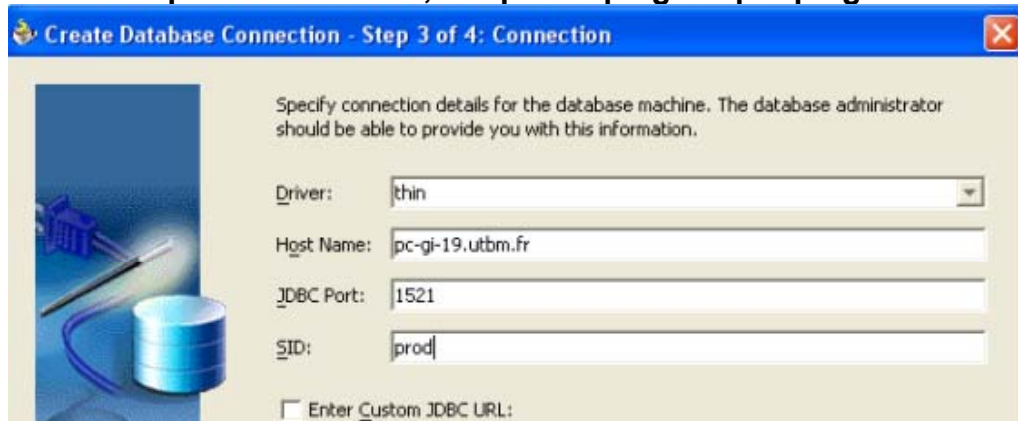
Création d'une procédure stockée avec Jdeveloper

supp_produit : permettant de supprimer un produit

Démarrer Jdeveloper 10G à partir du raccourci de Jdeveloper 10G créé sur votre bureau.

A partir du dossier Databases, créer une nouvelle connexion à l'aide de l'assistant de création de connexion sur votre compte **votrenom_gescom**
Paramètres à utiliser dans l'assistant de création de connexion :

Dans la capture ci-dessous, remplacer pc-gi-19 par pc-gi-203



Tester votre connexion.

Attention la connexion n'est pas conservée, il faut la recréer à chaque démarrage de Jdeveloper

Dans l'arborescence de Jdeveloper, sélectionner Procédure, puis New.

Le nom de la procédure est : **supp_produit**

Dans l'éditeur d'unité de programme compléter le code PL/SQL à partir de l'exemple ci-dessous :

-- le mot clé *create or replace* n'est pas obligatoire dans Jdeveloper

PROCEDURE supp_produit

```
(v_prodfref IN produit.prod_ref%TYPE)
IS
BEGIN
DELETE from produit WHERE prod_ref = v_prodfref;
  IF SQL%NOTFOUND THEN
    RAISE_APPLICATION_ERROR(-20202,'Pas de produit supprimé.');
```

Compiler et enregistrer la procédure.

Tester la procédure dans l'environnement de Jdeveloper :

```
supp_produit('TVCO84');
```

| F | Création d'une fonction stockée avec Sql*Plus ou iSQL*Plus |
|---|---|
| | <p>Création de la fonction lib_produit permettant de retourner le libellé d'un produit. A partir de votre éditeur compléter le code PL/SQL à partir de l'exemple ci-dessous :</p> <pre> create or replace function lib_produit (v_prodf IN produit.prod_ref%TYPE) return varchar2 IS lib_prod varchar2(30) ; BEGIN select PROD_DES INTO lib_prod from produit where PROD_REF=v_prodf; return (lib_prod); END lib_produit; / </pre> |
| | <p>Exécuter la fonction dans SQL*Plus :</p> <pre> SQL> variable v_lib varchar2(30); SQL> begin 2> :v_lib := lib_produit('TVCO84') ; 3> end ; 4> / </pre> |
| | <p>Vérifier l'exécution de votre procédure en affichant la valeur du résultat :</p> <pre> SQL> print v_lib ; V_LIB ----- <i>Television Couleur 84 cm 4/3</i> </pre> |
| | <p>Exécuter de nouveau la fonction avec une référence de produit inexistante :</p> <pre> SQL> begin 2> :v_lib := lib_produit('TVC100') ; 3> end ; 4> / </pre> <p>Que constatez-vous ? Le message d'erreur suivant est généré. <i>ERROR at line 1:</i> <i>ORA-01403: no data found</i> <i>ORA-06512: at "DEMOPL.LIB_PRODUI", line 7</i> <i>ORA-06512: at line 2</i></p> |

Modifier le code de votre fonction en intégrant un traitement d'exception :

create or replace function lib_produit

(v_prodfref IN produit.prod_ref%TYPE)

return varchar2

IS

lib_prod varchar2(30) := NULL;

BEGIN

select PROD_DES INTO lib_prod from produit where PROD_REF=v_prodfref;

return (lib_prod);

Exception

when no_data_found then return ('Produit inconnu');

END lib_produit;

/

Tester la nouvelle version de fonction et vérifier l'exécution de votre procédure en affichant la valeur du résultat :

SQL> begin

2> :v_lib := lib_produit('TVC100');

3> end ;

4> /

SQL> print v_lib ;

V_LIB

Produit inconnu

Afficher la liste des variables hôtes déclarées :

SQL> var

variable montant

datatype NUMBER

variable v_lib

datatype VARCHAR2(30)

G

Création d'une fonction stockée avec SQL Developer

Création de la fonction valider_produit permettant de vérifier l'existence de la référence d'un produit.

A partir de SQL Developer créer une nouvelle fonction stockée.

Le nom de la fonction est : valider_produit

Le type de la donnée retournée est de type numérique.

Dans l'éditeur d'unité de programme compléter le code PL/SQL à partir de l'exemple ci-dessous :

```
CREATE OR REPLACE FUNCTION valider_produit
(v_prodref IN produit.prod_ref%TYPE)
RETURN number IS
res number;
BEGIN
  SELECT 1 into res from produit WHERE prod_ref = v_prodref;
  IF SQL%FOUND
    THEN return 1 ;
  END IF ;
Exception
when no_data_found then return 0 ;
END;
```

Compiler et Enregistrer la fonction.

Exécuter la fonction pour le produit : TVCO84

The screenshot shows the 'Run PL/SQL' dialog box in SQL Developer. The 'Target' is set to 'VALIDER_PRODUIT'. The 'Parameters' table is as follows:

| Parameter | Data Type | Mode |
|----------------|-----------|------|
| <Return Value> | NUMBER | OUT |
| V_PRODREF | CHAR(6) | IN |

The PL/SQL Block editor shows the following code:

```
DECLARE
  V_PRODREF CHAR(6);
  v_Return NUMBER;
BEGIN
  V_PRODREF := 'TVCO84';

  v_Return := VALIDER_PRODUIT(
    V_PRODREF => V_PRODREF
  );
  DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
```

Exécuter de nouveau la procédure dans l'interpréteur avec un produit inexistant

Run PL/SQL

Target: VALIDER_PRODUIT

| Parameter | Data Type | Mode |
|----------------|-----------|------|
| <Return Value> | NUMBER | OUT |
| V_PRODREF | CHAR(6) | IN |

PL/SQL Block

```
DECLARE
  V_PRODREF CHAR(6);
  v_Return NUMBER;
BEGIN
  V_PRODREF := 'TVC100';

  v_Return := VALIDER_PRODUIT(
    V_PRODREF => V_PRODREF
  );
  DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
END;
```

Que constatez-vous ?

Tester la procédure à partir de **Sql*Plus**

1. Déclaration de la variable hôte

```
SQL > var existeprod number;
```

2. Exécution du bloc PL/SQL

```
SQL > begin
```

```
SQL> :existeprod := valider_produit('TVC100');
```

```
SQL>end ;
```

```
SQL> /
```

3. Vérification du résultat

```
SQL>print existeprod
```

Modifier la fonction afin qu'elle retourne une donnée de type booléen.

```
CREATE OR REPLACE FUNCTION valider_produit
(v_prodref IN produit.prod_ref%TYPE) return boolean
IS
res number;
BEGIN
  SELECT 1 into res from produit WHERE prod_ref = v_prodref;
  IF SQL%FOUND
    THEN return true ;
  END IF ;
Exception
  when no_data_found then return false;
END valider_produit;
/
```

Version 1**Création de la procédure inserer_lignecde qui permet d'ajouter une nouvelle ligne de commande dans la table LIGNE_COMMANDE.**

Spécifier des valeurs par défaut pour les différents paramètres :

La fonction valider_produit permettant de vérifier l'existence de la référence d'un produit devra être utilisée.

```

Create or replace PROCEDURE inserer_ligne_commande
(v_numcde IN LIGNE_COMMANDE.cde_num%TYPE      default 9999,
 v_prodef  IN LIGNE_COMMANDE.prod_ref%TYPE     default 'TVCO84',
 v_qte     IN LIGNE_COMMANDE.prod_qte%TYPE     default 0)
IS
BEGIN
-- la fonction valider_produit doit retourner un boolean
If valider_produit(v_prodef)
Then
    INSERT INTO LIGNE_COMMANDE (cde_num, prod_ref, prod_qte)
    VALUES (v_numcde, v_prodef, v_qte);
    COMMIT;
Else
    -- erreur de référence
    DBMS_OUTPUT.PUT_LINE ('Référence de produit incorrecte');
End if ;
END inserer_lignecde ;
/

```

Enregistrer la procédure.

Exécuter la procédure en insérant les lignes de commandes suivantes :

```
SQL> insert into commande values (3001, .... ) ;
```

```
SQL> execute inserer_ligne_commande(3001,'TVCO84',10);
PL/SQL procedure successfully completed.
```

```
SQL> select * from ligne_commande where cde_num=3000;
  CDE_NUM  PROD_R  PROD_QTE
-----
  3001  TVC84      10
```

```
SQL> set serveroutput on
```

```
SQL>execute inserer_ligne_commande(3001,'TVC100',5);
Référence de produit incorrecte
PL/SQL procedure successfully completed.
```

Version 2 :

Création de la procédure inserer_lignecde qui permet d'ajouter une nouvelle ligne de commande dans la table LIGNE_CDE.

Attention : La table LIGNE_CDE doit être créée à partir de la table LIGNE_COMMANDE et PRODUIT en incluant tous les champs nécessaires

Spécifier des valeurs par défaut pour les différents paramètres :
La fonction valider_produit permettant de vérifier l'existence de la référence d'un produit devra être utilisée.

```

Create or replace PROCEDURE inserer_ligne_cde
(v_numcde IN LIGNE_CDE.numcde%TYPE default 9999,
 v_numlig IN LIGNE_CDE.numlig%TYPE default 1,
 v_prodrf IN LIGNE_CDE.prodref%TYPE default 'TVCO84',
 v_prix IN LIGNE_CDE.prix%TYPE default 0,
 v_qte IN LIGNE_CDE.quantite%TYPE default 0)
IS
BEGIN
If valider_produit(v_prodrf)
Then
INSERT INTO LIGNE_CDE (numcde, numlig, prodref, prix, quantite)
VALUES (v_numcde, v_numlig, v_prodrf, v_prix, v_qte);
COMMIT;
Else
-- erreur de référence
DBMS_OUTPUT.PUT_LINE ('Référence de produit incorrecte');
End if ;
END inserer_lignecde ;
/

```

Enregistrer la procédure.

Exécuter la procédure en insérant les lignes de commandes suivantes :

```

SQL> execute inserer_lignecde(1,1,'TVCO84',7500,1);
PL/SQL procedure successfully completed.

```

```

SQL> select * from ligne_cde;

```

| NUMCDE | NUMLIG | PRODR | PRIX | QUANTITE |
|--------|--------|--------|------|----------|
| 1 | 1 | TVCO84 | 7500 | 1 |

```

SQL> set serveroutput on

```

```

SQL>execute inserer_lignecde(1,2,'TVC100',9900,1);

```

```

Référence de produit incorrecte
PL/SQL procedure successfully completed.

```

Étape 4 : Développement des packages et triggers

| | |
|---|--|
| I | Création d'un package paq_produit avec SQLDeveloper |
| Création de l'interface du package regroupant : Les procédures : Les fonctions : | |
| Création du corps du package regroupant : Les procédures : Les fonctions : | |

| | |
|---|--|
| J | Création des triggers de la base de données |
| A partir de WinDesign générer les triggers associés à votre modèle relationnel. | |

Étape 5 : Optimisation du modèle physique de données relationnel

| K | Optimisation du modèle physique de données |
|----------|---|
| | <p>1. A l'aide des éléments présentés dans le cours, proposez des solutions d'optimisation de votre modèle logique de données relationnel.</p> <p>Intégrer vos solutions dans le script SQL de création de votre application.</p> |
| | <p>2. <u>Validation de l'optimisation du modèle par l'écriture de requêtes.</u></p> <p>A partir du modèle initial rédiger les deux suivantes :</p> <ol style="list-style-type: none">1. Affichez la liste des clients (numéro et nom) de la catégorie « Détaillant » ayant commandé au cours du mois de mars.2. Affichez la liste des produits (référence et désignation) commandés par le client numéro 41103 depuis le début de l'année. <p>Si la structure de votre modèle a été modifiée, réécrire ces deux requêtes sur le modèle optimisé. Comparer la complexité et le temps d'exécution.</p> |
| | <p>3. <u>Mise en place de la gestion cohérente de la redondance</u></p> <p>A l'aide d'un trigger de mise à jour sur la table CATEGORIE, propager la mise à jour du libellé de catégorie sur cette colonne implantée de manière redondante dans la table CLIENT.</p> |

4. Etude de la stabilité des informations relatives aux prix.

Ecrire la requête SQL permettant de calculer le montant des commandes de janvier 1998.

Le résultat sera de la forme :

| CDE_NUM | DATE_CDE | MONTANT |
|---------|----------|---------|
| 18 | 01/01/98 | 7700 |
| 186 | 01/01/98 | 6200 |

...

Rédiger la requête permettant d'afficher le prix des produits des commandes 18 et 186.

Modifier le prix des produits :

Produit : F00255 passe de 600 à 630

Produit : P00783 passe de 400 à 450

Re-exécuter la requête de calcul des commandes de janvier 1998.

Quelle est la conséquence de la mise à jour sur votre modèle ?

Modifier votre base de données en conséquence.