Projet de BDA
Gestionnaire de vente de CD

Sébastien DERIVAUX
Guillaume AMOROS

Table des matières

1 Présentation

Le but du projet est de gérer les commandes d'une entreprise de ventes de CD par correspondance. L'application dispose de deux groupes de fonctionnalitées pour le client, le passage d'une commande et la consultation de ses commandes précédentes.

1.1 Passer des commandes

Les disques sont identifiés par un numéro de référence. On dispose des informations sur le nom de l'artiste, le titre de l'album, le genre musical auquel il appartient, le label éditeur, ainsi que son prix et la quantité en stock.

Le client est identifié par un pseudonyme, et on conserve également son nom, son prénom, son adresse et son numéro de carte bancaire. Un mot de passe lui permet d'accéder à ses données et ses commandes en cours.

On commence par demander si l'utilisateur possède déjà un compte. Si c'est le cas, on lui demande son pseudonyme et son mot de passe. Sinon, il doit renseigner les champs mentionnés précédemment. Il peut ensuite rechercher le ou les CD qu'il désire commander, d'après l'un des critères suivants : nom d'artiste, titre de l'album ou genre musical. Il peut choisir ensuite de commander un ou plusieurs CD, de relancer une recherche, modifier ses choix, les valider ou sortir (si une commande reste en cours elle sera conservée pour la prochaine connexion de l'utilisateur).

Listing 1: Exemple de recherches

```
1  -- Recherche par artiste se nommant John
2  SQL> CALL Gestion.RechercheArtiste('John');
3  Resultats
4  id | artiste | genre | label | prix | stock
5  2 | John Coltrane | Interstellar Space | Jazz |  | 15,75 | 44
6  3 | John Coltrane | Giant Step | Jazz |  | 15,75 | 6
7
8  -- Recherche par genre contenant la lettre 'a'.
9  SQL>  CALL Gestion.RechercheGenre('a');
10 Resultats
11 id | artiste | genre | label | prix | stock
12 1 | Bach | Offrande Musicale | Classique |  | 15,75 | 0
13 2 | John Coltrane | Interstellar Space | Jazz |  | 15,75 | 44
14 3 | John Coltrane | Giant Step | Jazz |  | 15,75 | 6
15 5 | Napalm Death | Utopia Banished | Death |  | 8,75 | 15
16 6 | Slayer | Decade of Aggression | Thrash |  | 20,75 | 30
```

Une commande est identifiée pas un numéro, et contient la date de validation, l'état de la commande (à valider, en cours ou terminée), la date d'envoi et l'identifiant du client qui l'a passée.

Lorsque le client commande un ou des CD, et qu'aucune commande "à valider" n'existe, une commande de ce type est crée et les CD ajoutés. Sinon ils sont simplement ajoutés à la commande "A valider". On les ajoute également à une table qui contient le détail de la commande, identifiée par le numéro de la commande, et qui contient le numéro du CD, le nombre commandé, et un état, par défaut "en attente".

Quand le client valide sa commande, on retire des stock les cd achetés si c'est possible. Dans ce cas, on édite la facture définitive, en ajoutant les frais de port, et la commande devient "en cours". Si un article n'est pas disponible en quantité suffisante, l'utilisateur doit modifier sa commande en conséquence. Si la commande est validé, les CD correspondant dans la table Détail Commande passent à l'état "disponible".

Listing 2: Exemple de passage de commande

```
1  -- L'utilisateur 'elmeira' commande le cd num 2 en 3 exemplaires
2  SQL> CALL Gestion.Ajoute('elmeira', 2, 3);
3
4  -- L'utilisateur 'elmeira' commande le cd num 3 en 4 exemplaires
5  SQL> CALL Gestion.CommandeAjoute('elmeira', 3,4);
6
7  -- L'utilisateur 'elmeira' affiche sa commande active
8  SQL> CALL Gestion.CommandeAffiche('elmeira');
9  cd | artiste | album | prix | qte
10 3 | John Coltrane | Giant Step | 15,75 | 4
11 2 | John Coltrane | Interstellar Space | 15,75 | 1
12 Total de votre commande: 82,75
13
14 -- Modification de nombre de cd num 2 acheté
15 SQL> CALL Gestion.CommandeModifie('elmeira', 2, 6);
16
17 -- Validation de la commande, pas de problème.
18 SQL> CALL Gestion.CommandeValide('elmeira');
19 cd | artiste | album | prix | qte
20 3 | John Coltrane | Giant Step | 15,75 | 4
21 2 | John Coltrane | Interstellar Space | 15,75 | 6
22 Total de votre commande: 164
23
24 -- Exemple d'une validation qui ne marche pas par manque de stocks
25 SQL>  CALL Gestion.CommandeValide('elmeira');
26 cd | artiste | album | prix | qte
27 1 | Bach | Offrande Musicale | 15,75 | 555
28 Pas assez de stock pour le cd: 1 il ne reste que 20
29 Veuillez modifier votre commande
30
31 -- Modification du nombre de cd acheté.
32 SQL> CALL Gestion.CommandeModifie('elmeira', 1, 20);
33
34 -- La validation fonctionne à présent.
35 SQL> CALL Gestion.CommandeValide('elmeira');
36 cd | artiste | album | prix | qte
37 1 | Bach | Offrande Musicale | 15,75 | 20
38 Total de votre commande: 326,5
```

1.2 Gestion de l'historique

On dispose aussi de fonctions pour gérer l'accès d'un client à l'historique de ses commandes. Le programme affiche la liste des commandes ainsi que leurs états. Le client peut visualiser le détail d'une commande précise.

Listing 3: Exemple de consultation d'historique

```
1  -- Consultation de l'historique du client seb
2  SQL> call Gestion.ConsulteHistorique('seb');
3  Consultation de l historique du client seb
4  ID | Etat | Validation | Envoie
5  2 | En cours | 06/04/04 |
6  3 | En cours | 06/04/04 |
7  4 | A valider |  |
8
9  -- Consultation de la commande 3 du client seb
10 SQL>  call Gestion.ConsulteHistorique('seb', 3);
11 Consultation de la commande 3 passee par le client seb
12 cd | artiste | album | prix | qte
13 1 | Bach | Offrande Musicale | 15,75 | 10
14 5 | Napalm Death | Utopia Banished | 8,75 | 5
15
16 -- Consultation de la commande 4 du client seb
17 SQL> call Gestion.ConsulteHistorique('seb', 4);
18 Consultation de la commande 4 passee par le client seb
19 cd | artiste | album | prix | qte
20 6 | Slayer | Decade of Aggression | 20,75 | 5
21 2 | John Coltrane | Interstellar Space | 15,75 | 1
22 4 | Napalm Death | Scum | 16,75 | 3
```

2 Aspects techniques

2.1 Curseurs

Afin de factoriser les procédure de recherche, nous utilisons des références de curseur, ce sont des curseur mais qui n'ont pas d'expression select associé. On peut s'en servir pour les passer à d'autre procédure.

Dans notre cas chacune des 3 procédures de recherche (RechercheAlbum, RechercheArtiste et RechercheGenre) ouvre un ref curseur avec une expression select différente puis le passe à la procédure AfficheListeCD qui s'occupe d'itérer le curseur et d'afficher les cd qu'il contient. A aucun momment AfficheListeCD ne sait, ni ne cherche à savoir, l'expression select qui a été utilisée. Le type de ref curseur (CDCurType ici) définit juste le format des données retournée.

Outre ces curseurs par référence, nous utilisons des curseurs explicites et implicites. Nous utilisons un curseur implicite dans le `FOR` de la procédure `CommandeAffiche` (ligne 251). Procéder ainsi permet d'être conçis et très clair.

2.2 Séquences

Les tables CD et Commande ont besoin d'un nombre unique pour leurs clefs primaires. Ce problème est généralement réglé en utilisant des séquences pour générer des nombres différents, c'est le choix qui a été fait. Une autre solution consiste à prendre le plus grand index dans table augmenté de un, mais cette solution est néanmoins plus coûteuse en ressources.

Nous avions d'abord un trigger à l'insertion qui s'occupe de remplir le nuplet avec la clé, mais cette méthode est déconseillé par Oracle, car elle fait chuter les performances et augmente la complexité sans raisons valables.

Listing 4: Gestion de la clé de la table CD dans un trigger

```
1 CREATE OR REPLACE TRIGGER OnInsertCD
2 BEFORE INSERT ON CD
3 FOR EACH ROW
4 WHEN (:new.cdId IS NULL)
5 BEGIN
6   SELECT cdId_seq.NEXTVAL
7   INTO   :new.cdId
8   FROM   dual;
9 END;
```

2.3 Exceptions

Il a été défini une exception d'application sous le code -20000 (les codes d'exception d'application allant de -20000 à -29999). Cette exception est lancée dans la fonction CommandeCourante qui renvoie l'identifiant de la commande en cours d'un client. Si une telle commande n'existe pas, elle appelle `raise_application_error` du package `dbms_standart` pour transformer l'exception `NO_DATA_FOUND` en notre exception nommée `expPasDeCommande`.

En plus de cette exception d'application, nous gérons l'exception de violation d'une contrainte de type `FOREIGN KEY` (code `ORA-02291`) et l'exception de violation de contrainte de type `CHECK` (code `ORA-02290`) que nous nommons respectivement `expForeignKey` et `expCheck`.

2.4 Validation de commande

La procédure de validation de commande étant la plus complexe, nous allons l'étudier en détail. L'objectif est de faire en sorte que la validation d'une commande soit atomique ; soit tout les articles sont validés soit aucun. Nous commençons donc par placer un point de sauvegarde. Ensuite, nous passons en revu tout les articles de la commande. Nous utilisons la clause `FOR UPDATE OF CD.cdStock` afin de nous assurer que personne ne modifie le stock des cd que nous allons retiré du magasin avant qu'ils ne soient effectivement retiré (par le commit). Il faut évidemment éviter qu'un autre client retire des cd entre le momment où nous avons tester qu'il reste assez de cd et le momment où nous les enlevons vraiment. Si la mise à jour de la quantité d'un cd provoque une exception sur la contrainte qu'un stock de cd doit rester positif, c'est qu'il n'y a pas assez de cd en stock et donc nous annulons toute la validation par le `ROLLBACK TO debut` en affichant l'article qui fait défaut.

3 Conclusion

Durant ce projet nous avons pu mettre en oeuvre la technologie PL/SQL d'Oracle. Ce language de 3^e génération sait se montrer très efficace dans son domaine. Nous avons aussi pu nous rendre compte de la bibliothèque fournie par Oracle pour son produit phare, notemment Application Developper's Guide – Fundamentals et PL/SQL – User's Guide and Reference. Il nous a été cependant très difficile de trouver de la documentation sur les types ou la syntaxes des requêtes (DML et DDL) à utiliser pour obtenir de meilleures performances.

Il nous reste à présent à développer une application java utilisant la base de donnée via JDBC afin d'obtenir un programme client qui va permettre de gérer la base de données de façons plus conviviale.

A Création des tables

Listing 5: tables.sql

```
1  -- Table Client
2  create table Client(
3    clId VARCHAR2(16) PRIMARY KEY,
4    clNom VARCHAR2(16),
5    clPrenom VARCHAR2(16),
6    clMpass VARCHAR2(16) NOT NULL,
7    clAdresse VARCHAR2(64),
8    clNumCarte NUMBER(16) NOT NULL
9  );
10
11 -- Données Client
12 INSERT INTO client VALUES ( 'seb', 'DERIVAUX', 'Sebastien', 'passe1',
13   'Wantzenau 67000 STRASBOURG', 6543212345678987 );
14 INSERT INTO client VALUES ( 'gnome', '', 'Guillaume', 'passe2',
15   'Espla 67000 STRASBOURG', 7654321234567898 );
16 INSERT INTO client VALUES ( 'claire', 'BAEGERT', 'CLAIRE', 'passe3',
17   '20, rue Albert Schweitzer 67000 STRASBOURG', 8987654321234567 );
18 INSERT INTO client VALUES ( 'elmeira', 'MEIRA', 'DAVID', 'passe4',
19   'Rsidence Universitaire 67000 STRASBOURG', 9876543212345678 );
20 INSERT INTO client VALUES ( 'peugeot', 'TEISSEIRE', 'RENAUD', 'passe5',
21   'Quartier juif 67000 STRASBOURG', 7654321234567898 );
22
23 -- Table Client
24 create table Genre(
25   genre VARCHAR2(16) PRIMARY KEY
26 );
27
28 -- Genre de musique autorisé
29 INSERT INTO Genre VALUES('Classique');
30 INSERT INTO Genre VALUES('Jazz');
31 INSERT INTO Genre VALUES('Death');
32 INSERT INTO Genre VALUES('Thrash');
33 INSERT INTO Genre VALUES('Grind');
34
35 -- Tables des cds.
36 create table CD(
37   cdId NUMBER PRIMARY KEY,
38   cdArtiste VARCHAR2(32) NOT NULL,
39   cdAlbum VARCHAR2(32) NOT NULL,
40   cdGenre VARCHAR2(16) NOT NULL,
41   cdLabel VARCHAR2(32),
42   cdPrix NUMBER(5,2) NOT NULL,
43   cdStock NUMBER(38) DEFAULT 0,
44   UNIQUE(cdArtiste, cdAlbum),
45   -- Une quantitée est toujours positive
46   CHECK(cdStock>=0),
47   FOREIGN KEY(cdGenre) REFERENCES Genre(genre)
48 );
49
50 -- Séquence génératrice de la clé de la table cd.
51 CREATE SEQUENCE cdId_seq;
52
53 -- Quelques cds.
54 INSERT INTO CD VALUES(cdId_seq.nextval,
55   'Bach', 'Offrande Musicale', 'Classique', NULL, 15.75, 50);
56 INSERT INTO CD VALUES(cdId_seq.nextval,
57   'John Coltrane', 'Interstellar Space', 'Jazz', NULL, 15.75, 50);
58 INSERT INTO CD VALUES(cdId_seq.nextval,
59   'John Coltrane', 'Giant Step', 'Jazz', NULL, 15.75, 10);
```

```
60 INSERT INTO CD VALUES(cdId_seq.nextval,
61    'Napalm Death', 'Scum', 'Grind', NULL, 16.75, 10);
62 INSERT INTO CD VALUES(cdId_seq.nextval,
63    'Napalm Death', 'Utopia Banished', 'Death', NULL, 8.75, 30);
64 INSERT INTO CD VALUES(cdId_seq.nextval,
65    'Slayer', 'Decade of Aggression', 'Thrash', NULL, 20.75, 30);
66
67 -- La table des commandes.
68 create table Commande(
69    cmId NUMBER PRIMARY KEY,
70    cmDateSoum DATE,
71    cmEtat VARCHAR2(10) DEFAULT 'A valider' NOT NULL,
72    cmDateEnvoie DATE,
73    cmClient VARCHAR2(16),
74    CHECK( cmEtat IN('A valider', 'En cours', 'Terminee')),
75    FOREIGN KEY (cmClient) REFERENCES Client (clId) ON DELETE CASCADE
76 );
77
78 -- Séquence génératrice de la clé des commandes.
79 CREATE SEQUENCE cmId_seq;
80
81 -- Les articles commandé dans les commandes.
82 create table DetailCmd(
83    dcCmd NUMBER,
84    dcCD NUMBER,
85    dcQte NUMBER(38) NOT NULL,
86    dcEtat VARCHAR2(10) DEFAULT 'En attente' NOT NULL,
87    PRIMARY KEY(dcCmd, dcCD),
88    CHECK(dcEtat IN('Disponible', 'En attente')),
89    -- Une quantitée est toujours positive
90    CHECK(dcQte>0),
91    FOREIGN KEY (dcCmd) REFERENCES Commande (cmId) ON DELETE CASCADE,
92    FOREIGN KEY (dcCD) REFERENCES CD (cdId)
93 );
```

B Suppression de la base de donnée

Listing 6: delete.sql

```
1 DROP TABLE Client CASCADE CONSTRAINT;
2 DROP TABLE Genre CASCADE CONSTRAINT;
3 DROP TABLE CD CASCADE CONSTRAINT;
4 DROP SEQUENCE cdId_seq;
5 DROP TABLE Commande CASCADE CONSTRAINT;
6 DROP SEQUENCE cmId_seq;
7 DROP TABLE DetailCmd CASCADE CONSTRAINT;
```

C Package de gestion

Listing 7: Gestion.sql

```
1  set serveroutput on;
2
3  CREATE OR REPLACE PACKAGE Gestion
4  AS
5     -- Ajout d'un client a la base de donnee
6     PROCEDURE AjoutClient(ident VARCHAR2, nom VARCHAR2, prenom VARCHAR2,
7                           mpass VARCHAR2, adresse VARCHAR2, numCarte NUMBER);
8     -- Ajout d'un CD a la base de donnee
9     PROCEDURE AjoutCD(artiste VARCHAR2, album VARCHAR2, genre VARCHAR2,
10                      label VARCHAR2, prix NUMBER, stock NUMBER);
11    -- Fonction de verification de l'identite d'un client
12    FUNCTION Identification(ident VARCHAR2, pass VARCHAR2) RETURN BOOLEAN;
13    -- Recherche par artiste
14    PROCEDURE RechercheArtiste(pattern VARCHAR2);
15    -- Recherche par album
16    PROCEDURE RechercheAlbum(pattern VARCHAR2);
17    -- Recherche par genre
18    PROCEDURE RechercheGenre(pattern VARCHAR2);
19
20
21
22    -- Ajout d'un article a une demande (quitte a creer la demande).
23    PROCEDURE CommandeAjoute(ident VARCHAR2, cd NUMBER, qte NUMBER);
24    -- Abandon de la commande en cours.
25    PROCEDURE CommandeAbandon(ident VARCHAR2);
```

```
26    -- Modification d'une quantitee d'un article.
27    PROCEDURE CommandeModifie(ident VARCHAR2, cd NUMBER, nvQte NUMBER);
28    -- Abandonne une article d'une commande.
29    PROCEDURE CommandeAbandon(ident VARCHAR2, cd NUMBER);
30    -- Affiche le contenu de la commande en cours.
31    PROCEDURE CommandeAffiche(ident VARCHAR2);
32    -- Valide la commande et sort la facture.
33    PROCEDURE CommandeValide(ident VARCHAR2);
34
35    -- Permet de consulter l'historique d'un client.
36    PROCEDURE ConsulteHistorique(ident VARCHAR2);
37    -- Permet de consulter une commande d'un client.
38    PROCEDURE ConsulteHistorique(ident VARCHAR2, cmd NUMBER);
39
40  END Gestion;
41  /
42
43  CREATE OR REPLACE  PACKAGE BODY Gestion
44  AS
45
46    -- Definie le type de curseur qui contient des nuplets de la table CD.
47    TYPE CDCurType IS REF CURSOR RETURN CD%ROWTYPE;
48
49    -- On definit certaines exceptions
50
51    -- Exception de violation de la contrainte de non referencement
52    expForeignKey EXCEPTION;
53    PRAGMA EXCEPTION_INIT(expForeignKey, -02291);
54
55    -- Exception de violation de la contrainte CHECK
56    expCheck EXCEPTION;
57    PRAGMA EXCEPTION_INIT(expCheck, -02290);
58
59    -- Exception si on tente de manipuler une commande courante inexistante.
60    -- Code defini par l'application.
61    expPasDeCommande EXCEPTION;
62    PRAGMA EXCEPTION_INIT(expPasDeCommande, -20000);
63
64  PROCEDURE AjoutClient(ident VARCHAR2, nom VARCHAR2, prenom VARCHAR2,
65                        mpass VARCHAR2, adresse VARCHAR2, numCarte NUMBER)
66  AS
67  BEGIN
68  INSERT INTO Client VALUES
69           (ident, nom, prenom, mpass, adresse, numCarte);
70  COMMIT;
71  EXCEPTION  -- Si on viole l'unicite on affiche un message
72        WHEN DUP_VAL_ON_INDEX THEN
73           dbms_output.put_line('Erreur, cette personne existe deja');
74  END;
75
76
77  PROCEDURE AjoutCD(artiste VARCHAR2, album VARCHAR2, genre VARCHAR2,
78                    label VARCHAR2, prix NUMBER, stock NUMBER)
79  AS
80    CURSOR c IS SELECT cdId FROM CD WHERE
81       cdArtiste=artiste AND cdAlbum=album AND cdGenre=genre
82      AND ((cdLabel IS NULL AND label IS NULL) OR cdLabel=label) AND prix=prix;
83    id NUMBER;
84    stock_act NUMBER;
85  BEGIN
86    OPEN c;
87    FETCH c INTO id;
88    IF c%NOTFOUND THEN
89      -- Si le cd n'existe pas encore on l'ajoute
90      INSERT INTO CD
91        VALUES (cdId_seq.nextval, artiste, album, genre, label, prix, stock);
92    ELSE
93      -- Sinon on met a jour sa quantite
94      UPDATE CD SET cdStock=cdStock+stock WHERE cdId=id;
95    END IF;
96    COMMIT;
97  END;
98
99
100
101 FUNCTION Identification(ident VARCHAR2, pass VARCHAR2) RETURN BOOLEAN
102 AS
```

```
103     nb NUMBER;
104   BEGIN
105     SELECT count(clId) INTO nb  FROM Client WHERE clId=ident AND clMpass=pass;
106     -- Si un nuplet correspond a ce login et ce password alors c'est que c'ets bon
107     -- Il ne peut pas y en avoir plus de toute facon
108     RETURN  nb=1;
109   END;
110
111
112
113   -- Procedure privee
114   -- Affiche des nuplet de la table CD
115   PROCEDURE AfficheListeCD(cur IN OUT CDCurType)
116   AS
117     it CD%ROWTYPE;
118
119   BEGIN
120     dbms_output.put_line('Resultats');
121     dbms_output.put_line(' id | artiste | genre | label | prix | stock');
122
123     -- le curseur est ouvert dans la fonction appelante.
124
125     LOOP
126       FETCH cur INTO it;
127       EXIT WHEN cur%NOTFOUND;
128       dbms_output.put_line(it.cdId || ' | ' ||
129                            it.cdArtiste || ' | ' ||
130                            it.cdAlbum || ' | ' ||
131                            it.cdGenre || ' | ' ||
132                            it.cdLabel || ' | ' ||
133                            it.cdPrix || ' | ' ||
134                            it.cdStock);
135     END LOOP;
136     CLOSE cur;
137   END;
138
139
140   PROCEDURE RechercheArtiste(pattern VARCHAR2)
141   AS
142     cur CDCurType;
143   BEGIN
144     OPEN cur FOR SELECT *
145         FROM CD WHERE LOWER(cdArtiste) like '%' || LOWER(pattern) || '%';
146     AfficheListeCD(cur);
147   END;
148
149   PROCEDURE RechercheAlbum(pattern VARCHAR2)
150   AS
151     cur CDCurType;
152   BEGIN
153     OPEN cur FOR SELECT *
154         FROM CD WHERE LOWER(cdAlbum) like '%' || LOWER(pattern) || '%';
155     AfficheListeCD(cur);
156   END;
157
158   PROCEDURE RechercheGenre(pattern VARCHAR2)
159   AS
160     cur CDCurType;
161   BEGIN
162     OPEN cur FOR SELECT *
163         FROM CD WHERE LOWER(cdGenre) like '%' || LOWER(pattern) || '%';
164     AfficheListeCD(cur);
165   END;
166
167   FUNCTION CommandeCourante(ident VARCHAR2) RETURN NUMBER
168   AS
169     retour NUMBER;
170   BEGIN
171     SELECT cmId INTO retour FROM Commande
172       WHERE cmClient=ident AND cmEtat='A valider';
173     RETURN retour;
174   EXCEPTION
175     WHEN NO_DATA_FOUND THEN
176       -- On balance une exception d'application cf expPasDeCommande
177       raise_application_error(-20000, 'Pas de commande ouverte pour '
178         || ident);
179   END;
```

```
180
181
182 PROCEDURE CommandeAjoute(ident VARCHAR2, cd NUMBER, qte NUMBER)
183 AS
184   cmd Commande.cmId%TYPE;
185   nb NUMBER;
186 BEGIN
187
188
189   -- On cherche l'id de la commande ouverte.
190   BEGIN
191     cmd:=CommandeCourante(ident);
192   EXCEPTION
193     -- Si ca lance une exception, c'est qu'il n'y avait pas commande courante
194     -- Il faut donc la creer.
195     WHEN expPasDeCommande THEN
196       INSERT INTO Commande(cmId, cmClient)
197         VALUES (cmId_seq.nextval, ident);
198       cmd:=CommandeCourante(ident);
199   END;
200
201
202
203   -- On va tenter d'inserer l'article, mais s'il y est deja pour cette
204   -- commande, on affiche juste un message d'erreur.
205   BEGIN
206     INSERT INTO DetailCmd(dcCmd, dcCD, dcQte) VALUES (cmd, cd, qte);
207   EXCEPTION
208     WHEN DUP_VAL_ON_INDEX THEN
209       dbms_output.put_line('Article deja present');
210   END;
211
212   COMMIT;
213 END;
214
215 PROCEDURE CommandeAbandon(ident VARCHAR2)
216 AS
217 BEGIN
218   -- On cherche l'id de la commande ouverte.
219   DELETE FROM Commande WHERE cmClient=ident AND cmEtat='A valider';
220   COMMIT;
221 END;
222
223 PROCEDURE CommandeModifie(ident VARCHAR2, cd NUMBER, nvQte NUMBER)
224 AS
225   cmd NUMBER;
226 BEGIN
227   cmd:=CommandeCourante(ident);
228   UPDATE DetailCmd SET dcQte=nvQte WHERE dcCD=cd AND dcCmd=cmd;
229   COMMIT;
230 END;
231
232 PROCEDURE CommandeAbandon(ident VARCHAR2, cd NUMBER)
233 AS
234   cmd NUMBER;
235 BEGIN
236   cmd:=CommandeCourante(ident);
237   DELETE DetailCmd WHERE dcCD=cd AND dcCmd=cmd;
238   COMMIT;
239 END;
240
241 PROCEDURE CommandeAffiche(ident VARCHAR2)
242 AS
243   total NUMBER:= 0;   -- Contriendra le cout total de la commande
244   totalCD NUMBER:= 0; -- Contiendra le nombre total de cd de la commande
245   cmdCourante NUMBER;
246 BEGIN
247   dbms_output.put_line('cd | artiste | album | prix | qte');
248
249   cmdCourante := CommandeCourante(ident);
250
251   FOR it IN
252   (  -- curseur implicite
253     SELECT cdId, cdArtiste, cdAlbum, cdPrix, dcQte
254       FROM DetailCmd, CD
255       WHERE dcCmd=cmdCourante AND cdId=dcCD
256   )
```

```
257     LOOP
258       dbms_output.put_line( it.cdId || ' | ' ||
259                             it.cdArtiste || ' | ' ||
260                             it.cdAlbum || ' | ' ||
261                             it.cdPrix || ' | ' ||
262                             it.dcQte);
263       total := total + it.cdPrix * it.dcQte;
264       totalCD := totalCD + it.dcQte;
265     END LOOP;
266
267     IF total=0 THEN
268       dbms_output.put_line('Votre commande est vide.');
269     ELSE
270       -- On ajoute les frais de port
271       total := total + 2;
272       total := total + 0.5 * (totalCD - 1);
273       dbms_output.put_line('Total de votre commande: ' || total);
274     END IF;
275   END;
276
277   PROCEDURE CommandeValide(ident VARCHAR2)
278   AS
279     total NUMBER:= 0;    -- Contriendra le cout total de la commande
280     totalCD NUMBER:= 0; -- Contiendra le nombre total de cd de la commande
281     cmdCourante NUMBER;
282   BEGIN
283     dbms_output.put_line('cd | artiste | album | prix | qte');
284
285     cmdCourante := CommandeCourante(ident);
286
287
288     -- Si il y a un probleme on reviendras dans l'etat de la BD comme ici.
289     SAVEPOINT debut;
290
291     -- Mise a jour de la commande
292     FOR it IN
293     (  -- Curseur implicite
294       SELECT cdId, cdArtiste, cdAlbum, cdPrix, dcQte, cdStock
295         FROM DetailCmd, CD
296         WHERE dcCmd=cmdCourante AND cdId=dcCD
297         -- On utilise for update pour que personne ne touche la table des cd
298         -- avant le commit
299         FOR UPDATE OF CD.cdStock
300     )
301     LOOP
302     BEGIN
303       dbms_output.put_line( it.cdId || ' | ' ||
304                             it.cdArtiste || ' | ' ||
305                             it.cdAlbum || ' | ' ||
306                             it.cdPrix || ' | ' ||
307                             it.dcQte);
308       total := total + it.cdPrix * it.dcQte;
309       totalCD := totalCD + it.dcQte;
310       UPDATE CD SET cdStock=cdStock-it.dcQte WHERE cdId=it.cdId;
311     EXCEPTION
312       -- A priori on a juste l'exception de passer un stock sous 0.
313       WHEN expCheck THEN
314         dbms_output.put_line('Pas assez de stock pour le cd: ' || it.cdId
315           || ' il ne reste que ' || it.cdStock);
316         RAISE; -- On relance l'exception
317     END;
318     END LOOP;
319
320     UPDATE Commande SET cmEtat='En cours', cmDateSoum=SYSDATE WHERE cmID=cmdCourante;
321     UPDATE DetailCmd SET dcEtat='Disponible' WHERE dcCmd=cmdCourante;
322
323     COMMIT;
324
325     IF total=0 THEN
326       dbms_output.put_line('Votre commande est vide.');
327     ELSE
328       -- On rajoute les frais de port
329       total := total + 2;
330       total := total + 0.5 * (totalCD - 1);
331       dbms_output.put_line('Total de votre commande: ' || total);
332     END IF;
333      EXCEPTION
```

```
334       -- Si on a eut un prob de stock on fait le rollback.
335       WHEN expCheck THEN
336         dbms_output.put_line('Veuillez modifier votre commande');
337         ROLLBACK TO debut;
338       WHEN OTHERS THEN
339         ROLLBACK TO debut;
340         RAISE;  -- On ne traite pas une exception inconnue
341   END;
342
343   PROCEDURE ConsulteHistorique(ident VARCHAR2)
344   AS
345   BEGIN
346     dbms_output.put_line('Consultation de l historique du client '
347       || ident);
348
349     dbms_output.put_line('ID | Etat | Validation | Envoie');
350
351     FOR it IN
352     (  -- Curseur implicite
353       SELECT cmId, cmEtat, cmDateSoum, cmDateEnvoie
354         FROM Commande
355         WHERE cmClient=ident
356     )
357     LOOP
358       dbms_output.put_line( it.cmId || ' | ' ||
359                             it.cmEtat || ' | ' ||
360                             it.cmDateSoum || ' | ' ||
361                             it.cmDateEnvoie);
362     END LOOP;
363   END;
364
365   PROCEDURE ConsulteHistorique(ident VARCHAR2, cmd NUMBER)
366   AS
367   BEGIN
368     dbms_output.put_line('Consultation de la commande '  || cmd
369       || ' passee par le client ' || ident);
370
371     dbms_output.put_line('cd | artiste | album | prix | qte');
372
373     FOR it IN
374     (  -- Curseur implicite
375       SELECT cdId, cdArtiste, cdAlbum, cdPrix, dcQte
376         FROM DetailCmd, CD, Commande
377         WHERE dcCmd=cmd AND dcCmd=cmId AND cdId=dcCD AND cmClient=ident
378     )
379     LOOP
380       dbms_output.put_line( it.cdId || ' | ' ||
381                             it.cdArtiste || ' | ' ||
382                             it.cdAlbum || ' | ' ||
383                             it.cdPrix || ' | ' ||
384                             it.dcQte);
385     END LOOP;
386   END;
387
388   END Gestion;
389   /
```

D Jeu de test

Listing 8: Test2.sql

```
1   prompt ==> Test de login
2   prompt => Login valide 'seb'/'passe1'
3   BEGIN
4     IF Gestion.Identification('seb','passe1') THEN
5       dbms_output.put_line('Login Ok');
6     ELSE
7       dbms_output.put_line('Erreur de login');
8     END IF;
9   END;
10  /
11  pause
12  prompt => Login invalide 'pipo'/'gateau'
13  BEGIN
14    IF Gestion.Identification('pipo','gateau') THEN
15      dbms_output.put_line('Login Ok');
16    ELSE
```

```
17      dbms_output.put_line('Erreur de login');
18    END IF;
19  END;
20  /
21  pause
22  prompt
23
24
25
26  prompt ==> Test de recherches
27  prompt => Recherche par album 'a'
28  CALL Gestion.RechercheAlbum('a');
29  pause
30  prompt
31  prompt => Recherche par genre 'a'
32  CALL Gestion.RechercheGenre('a');
33  pause
34  prompt
35  prompt => Recherche par artiste 'John'
36  CALL Gestion.RechercheArtiste('John');
37  pause
38  prompt
39  prompt
40
41  prompt ==> Test de passage de commande pour Claire et affichage
42  prompt => Commande de 1 exemplaires du cd 2
43  CALL Gestion.CommandeAjoute('claire', 2, 2);
44  CALL Gestion.CommandeAffiche('claire');
45  pause
46  prompt
47  prompt => Commande de 3 exemplaires du cd 1 et affichage
48  CALL Gestion.CommandeAjoute('claire', 1, 3);
49  CALL Gestion.CommandeAffiche('claire');
50  pause
51
52  prompt
53  prompt => Tentative de validation
54  CALL Gestion.CommandeValide('claire');
55  prompt
56  pause
57
58  prompt => On retire le cd 1 de notre commande et on valide
59  CALL Gestion.CommandeAbandon('claire', 1);
60  CALL Gestion.CommandeValide('claire');
61  prompt
62  pause
63
64
65  prompt
66  prompt ==> Affichage de l'historique
67  prompt => Historique des commandes de 'seb'
68  CALL Gestion.ConsulteHistorique('seb');
69  prompt
70  pause
71  prompt => Detail de la commande 4 de seb
72  CALL Gestion.ConsulteHistorique('seb', 4);
73  prompt
74  pause
75
76  prompt ==> Mettre une note
```