
# Université Toulouse 1 Capitole

# MASTERS MIAGE
## PARCOURS
# IPM

Le raport de semaine bloquée de

## ORACLE : SQL et PL-SQL

**Yan ZHAO**
**Tianyuan LIU**

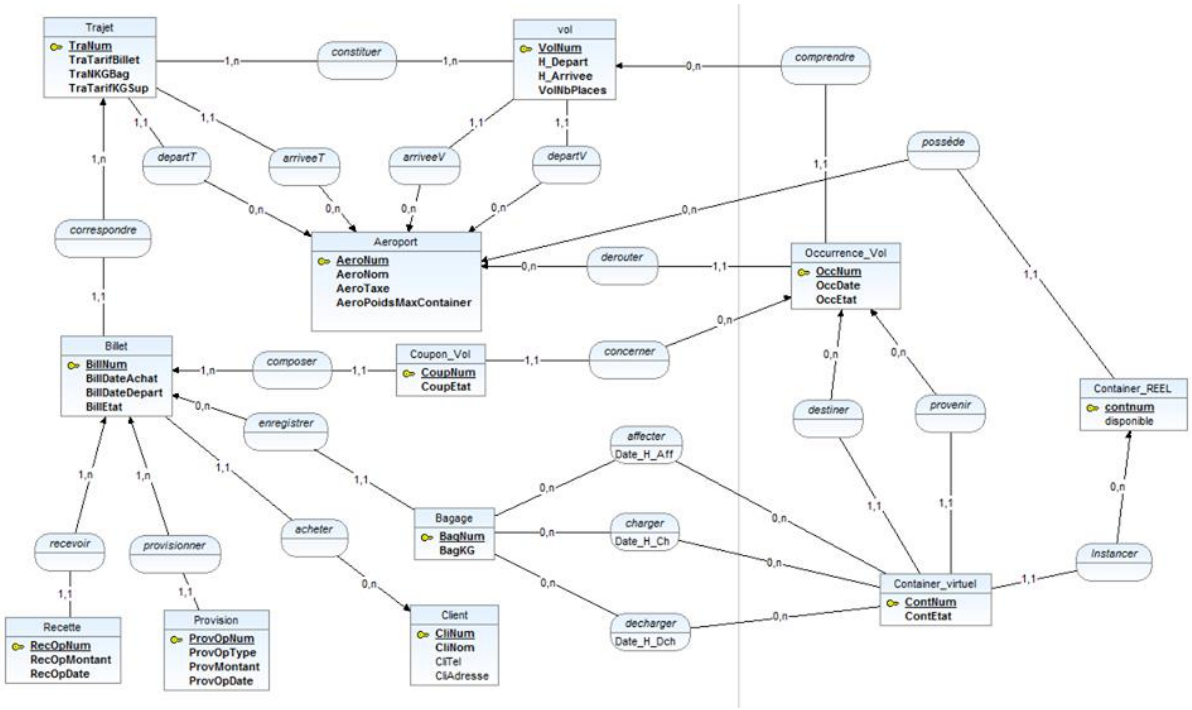
**15/01/2018**

# Table des matières

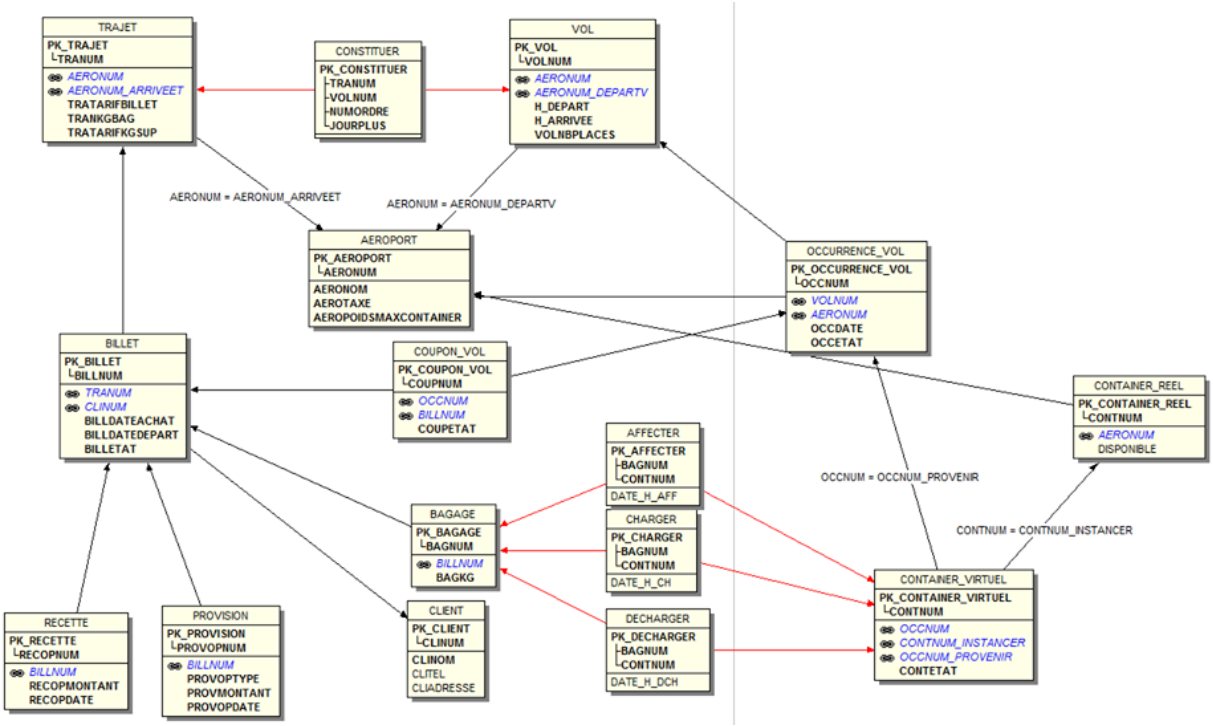
# Partie I : Création et administration de la base de données

## 1. Résultat de votre rétro-conception

### 1.1. MCD




### 1.2. MLD



## 2. Règles de gestion associées

Pour gérer le système d'AéroFrance, nous avons géré 5 grandes fonctionnalités : gestion de billets, gestion d'occurrences, gestion de bagages, gestion financière, gestion de conteneurs.

### 1. Gestion de billets

#### 1.1. GR_Achat de billets

Pour acheter un billet, le client doit saisir / choisir un trajet et une date de départ.

SI ce client n'a pas encore réservé un billet pour ce trajet à ce jour (un client peut juste acheter un billet pour un trajet pour un jour de départ), le système vérifie d'abord si toutes les occurrences correspondantes ont des sièges disponibles. Si un des occurrences de vol a déjà atteint son nombre maximum de réservation, l'action d'achet est échouée.

Si ce trajet dont toutes les occurrences de vol ont des sièges disponibles, le système crée un billet en état « émis » pour ce client et tous les coupons des occurrences de vol correspondantes sont créés automatiquement. Les états initiaux des coupons sont « réservé ».

*Référence : PRO_AJOUTER_BILLET, TRIG_INSERT_BILLET*

#### 1.2. GR_Gestion des états des billets

L'état initial d'un billet est « émis ». Lorsqu'un client embarque sur son premier vol, le billet passe dans l'état en cours. Lorsqu'un client arrive sur son dernier vol, le billet passe dans l'état terminé. Une fois l'état d'un billet passe à « terminé », il faut débiter sur le compte de provision et créditer sur le compte de recette (Voir GR_Gestion financier)

*Référence : PRO_AJOUTER_BILLET, PRO_RG_ARRIVE*

### 2. Gestion d'occurrences

#### 2.1. Gestion d'enregistrement

Quand l'état de l'occurrence du vol est <ouvert à l'embarquement>, des clients peuvent s'inscrire avec ses coupons. Une fois un enregistrement est fait, l'état du coupon concerné passe à <enregistré>. Si le coupon de l'occurrence de vol est le premier coupon d'un billet, l'état de ce billet passe à <En cours>.

Des clients peuvent s'inscrire avec ses bagages, si c'est le cas, on regarde les règles de Gestion de bagages.

*Référence :TRIG_ENREGISTREMENT*

#### 2.2. RG_Gestion des états d'occurrence de vol

Une occurrence de vol a plusieurs états possibles : ouverte à la réservation, ouverte à l'embarquement, ouverte à la liste d'attente, décollée, annulée, retardée, déroutée, arrivée.

a. Lorsque l'état passe à « décollée »
    a) Tous les bagages chargés sont automatiquement affectés à des conteneurs en vue de préparer leur déchargement dans l'aéroport de destination (regarder gestion d'affectation).
    b) Tous les coupons correspondants qui ne sont pas dans l'état enregistré, passent automatiquement dans l'état <annulé>.
    *Référence : PRO_RG_DECOLLEE*
b. Lorsque l'état passe à « arrivée»
    a) L'état des coupons de vol correspondants passe à <arrivé>.
    b) Pour chaque coupon correspondant, on regarde son ordre du billet. Si le coupon d'occurrence de vol est le dernier coupon du billet, l'état de ce billet passe à <terminé>
    *Référence : PRO_RG_ARRIVE*
c. Lorsque l'état passe à « déroutée»
    a) Tous les coupons correspondants passent automatiquement dans l'état <arrivé>.
    b) Des nouveaux coupons sont automatiquement émis sur la première occurrence de vol dans laquelle il reste des places et qui part de la destination sur laquelle a été dérouté l'avion pour arriver à la destination finale du vol dérouté.
    c) Passer les nouveaux coupons dans l'état <réservé>.
    d) Annuler l'affectation des bagages de l'ancien
    *Référence : PRO_RG_DECOUTEE*

## 3. Gestion de bagages

### 3.1. Enregistrement de bagages

Lorsqu'un bagage est enregistré, il faut comparer son poids avec le poids limité pour le trajet correspondant. Si c'est surchargé, on crédite sur le compte de provision le montant éventuellement payé par le client. Ensuite, l'affectation de ce bagage est exécutée automatiquement (Gestion d'affectation des bagages).

*Référence : PRO_RG_PROVISION_BAGAGE*

### 3.2. Gestion d'affectation des bagages

Le système vérifie les disponibilités de tous les conteneurs virtuels pour les occurrences de vol correspondantes. S'il n'y a aucun conteneur virtuel pour cette occurrence de vol, le système en crée un automatiquement. S'il y a un conteneur virtuel qui a la capacité à charger ce bagage, le système affecte ce bagage à ce conteneur virtuel. Si tous les conteneurs virtuels ne sont pas disponibles, le système crée automatiquement un nouvel conteneur virtuel et affect ce bagage à ce nouvel conteneur virtuel.

*Référence : PRO_RG_AFFECTER*

### 3.3. Gestion de chargement des bagages

Lorsque les employés scannent les codes-barres des bagages lors du chargement de ces derniers sur un conteneur

On regarde d'abord si ce bagage est affecté à ce conteneur, sinon, on affiche une erreur ; si oui, si c'est le premier bagage scanné pour une occurrence de vol, l'état de conteneur passe à « en cours de déchargement »

Si tous les bagages dans la liste d'affectation sont scannés, l'état de conteneur passe à « chargé ».

*Référence : PRO_CHARGER*

### 3.4. Gestion de déchargement des bagages

Quand les employés scannent les codes-barres des bagages lors du déchargement de ces derniers sur un conteneur, si c'est le premier bagage d'un conteneur, l'état de ce conteneur passe à « en cours de déchargement », si tous les bagages dans un conteneur sont déchargés, l'état de ce conteneur passe à « déchargement ».

*Référence : PRO_DECHARGER*

## 4. Gestion de conteneurs réel

Chaque aéroport a certains conteneurs pour transporter des bagages. On suppose que le poids max des conteneurs est la même pour chaque aéroport. La notion conteneur virtuel est  pour gérer l'historique d'affectations de bagages.

Lorsqu'un conteneur virtuel est créé, le système distribue automatiquement un conteneur réel à ce conteneur virtuel et la disponibilité de ce conteneur réel passe à « non ». Lorsqu'un conteneur virtuel passe à état en « déchargé », la disponibilité du conteneur réel correspondant  passe à « oui ».

*Référence : PRO_RG_AFFECTER*

## 5. Gestion financière

La gestion financière des billets s'effectue en plusieurs temps.

a. A l'émission d'un billet
   Le montant (montant du billet plus les taxes des différents aéroports par lesquels transite le trajet du billet) correspondant est crédité dans un compte de provision.
b. A l'enregistrement des bagages
   On crédite sur le compte de provision le montant pour la surcharge des bagages de ce client.
c. A la fin des billets
   Les deux montants au-dessus seront débités du compte de provision pour être crédités sur le compte de recette.

*Référence: PRO_RG_PROVISION_TICKET, PRO_RG_PROVISION_BAGAG, PRO_RG_ARRIVE*

## 3. Création des utilisateurs

```
-- L'utilisateur: Client
CREATE USER bestgroup_client IDENTIFIED BY "123"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP";
GRANT CREATE SESSION TO BESTGROUP_CLIENT;
--Donner le droit à executer le processus pour acheter un billet
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_AJOUTER_BILLET" TO "BESTGROUP_CLIENT" ;


-- l'utilisateur2: Contrôleur d'aeroport
CREATE USER bestgroup_controleur_aeroport IDENTIFIED BY "123"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP";
GRANT CREATE SESSION TO BESTGROUP_CONTROLEUR_AEROPORT;
--Donner le droit à executer le processus pour changer l'état de aeroport
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_CHANGER_OCCURRENCE_ETAT" TO "BESTGROUP_CONTROLEUR_AEROPORT";

-- l'utilisateur3: Bagagiste
CREATE USER bestgroup_bagagiste IDENTIFIED BY "123"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP";
GRANT CREATE SESSION TO BESTGROUP_BAGAGISTE;
--Donner le droit à executer le processus pour charger et décharger des bagages
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_CHARGER" TO "BESTGROUP_BAGAGISTE";
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_DECHARGER" TO "BESTGROUP_BAGAGISTE";

-- L'utilisateur4: Hôtesse d'accueil
CREATE USER bestgroup_hotesse IDENTIFIED BY "123"
DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP";
GRANT CREATE SESSION TO BESTGROUP_HOTESSE;
--Donner le droit à executer le processus pour l'embarquement et enregistrer le bagage du client
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_ENREGISTRER_COUPONVOL" TO "BESTGROUP_HOTESSE";
GRANT EXECUTE ON "DBA_PROJET_BESTGROUP"."PRO_AJOUTER_BAGAGE" TO "BESTGROUP_HOTESSE";
```

On définit 4 types de l'utilisateur correspondant le sujet :

1. Client : La personne qui peut exécuter le processus pour acheter le billet.

2. Contrôleur d'aéroport : La personne qui peut changer l'état de l'occurrence de vol.

3. Bagagiste : La personne qui peut charger et décharger les bagages du client.

4. Hôtesse d'accueil : La personne qui est chargé d'enregistrer le bagage du client et enregistrer le coupon de vol.

## 4. Création de l'environnement

Pour créer l'environnement, nous avons d'abord utilisé Win'Design pour générer une première version de scripte. Ensuite, nous avons modifié les types des données et jugé si les données peuvent être « null » ou pas.

Le script de la création des tables est en dessous :

1. Création des tables et ses clés prémaires / étrangère

```
-- ----------------------------------------------------------------------------
--       TABLE : COUPON_VOL
-- ----------------------------------------------------------------------------

CREATE TABLE COUPON_VOL
    (
     COUPNUM NUMBER  NOT NULL,
     OCCNUM NUMBER  NOT NULL,
     BILLNUM NUMBER  NOT NULL,
     COUPETAT VARCHAR2(50)  NOT NULL
,    CONSTRAINT PK_COUPON_VOL PRIMARY KEY (COUPNUM),
Constraint check_coupetat
  check (COUPETAT in ('enregistré','annulé','réservé','arrivée'))
     ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE COUPON_VOL
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_COUPON_VOL_OCCURRENCE_VOL
      ON COUPON_VOL (OCCNUM ASC)
     ;

CREATE  INDEX I_FK_COUPON_VOL_BILLET
      ON COUPON_VOL (BILLNUM ASC)
     ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : RECETTE
-- ----------------------------------------------------------------------------

CREATE TABLE RECETTE
   (
    RECOPNUM NUMBER  NOT NULL,
    BILLNUM NUMBER  NOT NULL,
    RECOPMONTANT NUMBER  NOT NULL,
    RECOPDATE DATE  NOT NULL
,   CONSTRAINT PK_RECETTE PRIMARY KEY (RECOPNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE RECETTE
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_RECETTE_BILLET
     ON RECETTE (BILLNUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : AEROPORT
-- ----------------------------------------------------------------------------

CREATE TABLE AEROPORT
   (
    AERONUM NUMBER  NOT NULL,
    AERONOM VARCHAR2(50)  NOT NULL,
    AEROTAXE NUMBER  NOT NULL,
    AEROPOIDSMAXCONTAINER NUMBER  NOT NULL
,   CONSTRAINT PK_AEROPORT PRIMARY KEY (AERONUM)
   ) ;

-- ----------------------------------------------------------------------------
--       TABLE : VOL
-- ----------------------------------------------------------------------------

CREATE TABLE VOL
   (
    VOLNUM NUMBER  NOT NULL,
    AERONUM_ARRIVEE NUMBER  NOT NULL,
    AERONUM_DEPART NUMBER  NOT NULL,
    H_DEPART DATE  NOT NULL,
    H_ARRIVEE DATE  NOT NULL,
    VOLNBPLACES NUMBER  NOT NULL
,   CONSTRAINT PK_VOL PRIMARY KEY (VOLNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE VOL
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_VOL_AEROPORT
     ON VOL (AERONUM_ARRIVEE ASC)
    ;

CREATE  INDEX I_FK_VOL_AEROPORT1
     ON VOL (AERONUM_DEPART ASC)
    ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : BILLET
-- ----------------------------------------------------------------------------

CREATE TABLE BILLET
    (
     BILLNUM NUMBER  NOT NULL,
     TRANUM NUMBER  NOT NULL,
     CLINUM NUMBER  NOT NULL,
     BILLDATEACHAT DATE  NOT NULL,
     BILLDATEDEPART DATE  NOT NULL,
     BILLETAT VARCHAR2(50)  NOT NULL
,    CONSTRAINT PK_BILLET PRIMARY KEY (BILLNUM)
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE BILLET
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_BILLET_TRAJET
     ON BILLET (TRANUM ASC)
    ;

CREATE  INDEX I_FK_BILLET_CLIENT
     ON BILLET (CLINUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : TRAJET
-- ----------------------------------------------------------------------------

CREATE TABLE TRAJET
    (
     TRANUM NUMBER  NOT NULL,
     AERONUM_DEPART NUMBER  NOT NULL,
     AERONUM_ARRIVEE NUMBER  NOT NULL,
     TRATARIFBILLET NUMBER(*,2)  NOT NULL,
     TRANKGBAG NUMBER  NOT NULL,
     TRATARIFKGSUP NUMBER  NOT NULL
,    CONSTRAINT PK_TRAJET PRIMARY KEY (TRANUM)
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE TRAJET
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_TRAJET_AEROPORT
     ON TRAJET (AERONUM_DEPART ASC)
    ;

CREATE  INDEX I_FK_TRAJET_AEROPORT1
     ON TRAJET (AERONUM_ARRIVEE ASC)
    ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : OCCURRENCE_VOL
-- ----------------------------------------------------------------------------

CREATE TABLE OCCURRENCE_VOL
    (
     OCCNUM NUMBER  NOT NULL,
     VOLNUM NUMBER  NOT NULL,
     AERONUM NUMBER  NULL,
     OCCDATE DATE NOT NULL,
     OCCETAT VARCHAR2(50) NULL
,    CONSTRAINT PK_OCCURRENCE_VOL PRIMARY KEY (OCCNUM),
Constraint check_onnetat
  CHECK (OCCETAT in ('ouverte à la réservation', 'ouverte à l''embarquement'
     ,' ouverte à la liste d''attente', 'décollée', 'annulée', 'retardée'
     , 'déroutée', 'arrivée'))
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE OCCURRENCE_VOL
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_OCCURRENCE_VOL_VOL
      ON OCCURRENCE_VOL (VOLNUM ASC)
     ;

CREATE  INDEX I_FK_OCCURRENCE_VOL_AEROPORT
      ON OCCURRENCE_VOL (AERONUM ASC)
     ;

-- ----------------------------------------------------------------------------
--       TABLE : CONTAINER_VIRTUEL
-- ----------------------------------------------------------------------------

CREATE TABLE CONTAINER_VIRTUEL
    (
     CONTNUM NUMBER  NOT NULL,
     OCCNUM_DESTINER NUMBER  NULL,
     OCCNUM_PROVENIR NUMBER  NULL,
     CONTETAT VARCHAR2(50)  NULL,
   CONTREEL NUMBER NULL
,    CONSTRAINT PK_CONTAINER PRIMARY KEY (CONTNUM),
Constraint check_CONTETAT
  CHECK (CONTETAT in (null,'en cours de chargement','chargé'
     ,'en cours de déchargement','déchargé'))
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE CONTAINER_VIRTUEL
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_CONTAINER_OCCURRENCE_VOL
      ON CONTAINER_VIRTUEL (CONTREEL ASC)
     ;
CREATE  INDEX I_FK_CONTAINER_CONTREEL
      ON CONTAINER_VIRTUEL (OCCNUM_DESTINER ASC)
     ;
CREATE  INDEX I_FK_CONTAINER_OCCURRENCE_VOL1
      ON CONTAINER_VIRTUEL (OCCNUM_PROVENIR ASC)
     ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : CONTAINER_REEL
-- ----------------------------------------------------------------------------
CREATE TABLE CONTAINER_REEL
(
  CONTNUM NUMBER NOT NULL,
  AERONUM NUMBER NOT NULL,
  DISPONIBLE varchar2(20) NOT NULL,
  CONSTRAINT PK_CONTAINER_REEL PRIMARY KEY (CONTNUM),
  Constraint check_DISPONIBLE
  CHECK (DISPONIBLE in ('oui','non'))
);
-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE CONTAINER_REEL
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_CONTREEL_OCCURRENCE_VOL
     ON CONTAINER_REEL (AERONUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : BAGAGE
-- ----------------------------------------------------------------------------

CREATE TABLE BAGAGE
   (
    BAGNUM NUMBER  NOT NULL,
    BILLNUM NUMBER  NOT NULL,
    BAGKG NUMBER  NOT NULL
,   CONSTRAINT PK_BAGAGE PRIMARY KEY (BAGNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE BAGAGE
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_BAGAGE_BILLET
     ON BAGAGE (BILLNUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : PROVISION
-- ----------------------------------------------------------------------------

CREATE TABLE PROVISION
   (
    PROVOPNUM NUMBER  NOT NULL,
    BILLNUM NUMBER  NOT NULL,
    PROVOPTYPE VARCHAR2(50)  NOT NULL,
    PROVMONTANT NUMBER  NOT NULL,
    PROVOPDATE DATE  NOT NULL
,   CONSTRAINT PK_PROVISION PRIMARY KEY (PROVOPNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE PROVISION
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_PROVISION_BILLET
     ON PROVISION (BILLNUM ASC)
    ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : CLIENT
-- ----------------------------------------------------------------------------

CREATE TABLE CLIENT
    (
     CLINUM NUMBER  NOT NULL,
     CLINOM VARCHAR2(50)  NOT NULL,
     CLITEL VARCHAR2(50)  NULL,
     CLIADRESSE VARCHAR2(50)  NULL
,    CONSTRAINT PK_CLIENT PRIMARY KEY (CLINUM)
    ) ;

-- ----------------------------------------------------------------------------
--       TABLE : DECHARGER
-- ----------------------------------------------------------------------------

CREATE TABLE DECHARGER
    (
     BAGNUM NUMBER  NOT NULL,
     CONTNUM NUMBER  NOT NULL,
     DATE_H_DCH DATE  NULL
,    CONSTRAINT PK_DECHARGER PRIMARY KEY (BAGNUM, CONTNUM)
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE DECHARGER
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_DECHARGER_BAGAGE
     ON DECHARGER (BAGNUM ASC)
    ;

CREATE  INDEX I_FK_DECHARGER_CONTAINER
     ON DECHARGER (CONTNUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : CONSTITUER
-- ----------------------------------------------------------------------------

CREATE TABLE CONSTITUER
    (
     TRANUM NUMBER  NOT NULL,
     VOLNUM NUMBER  NOT NULL,
     NUMORDRE NUMBER  NOT NULL,
     JOURPLUS NUMBER  NOT NULL
,    CONSTRAINT PK_CONSTITUER PRIMARY KEY (TRANUM, VOLNUM, NUMORDRE, JOURPLUS)
    ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE CONSTITUER
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_CONSTITUER_TRAJET
     ON CONSTITUER (TRANUM ASC)
    ;

CREATE  INDEX I_FK_CONSTITUER_VOL
     ON CONSTITUER (VOLNUM ASC)
    ;
```

```
-- ----------------------------------------------------------------------------
--       TABLE : CHARGER
-- ----------------------------------------------------------------------------

CREATE TABLE CHARGER
   (
    BAGNUM NUMBER  NOT NULL,
    CONTNUM NUMBER  NOT NULL,
    DATE_H_CH DATE  NULL
,   CONSTRAINT PK_CHARGER PRIMARY KEY (BAGNUM, CONTNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE CHARGER
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_CHARGER_BAGAGE
     ON CHARGER (BAGNUM ASC)
    ;

CREATE  INDEX I_FK_CHARGER_CONTAINER
     ON CHARGER (CONTNUM ASC)
    ;

-- ----------------------------------------------------------------------------
--       TABLE : AFFECTER
-- ----------------------------------------------------------------------------

CREATE TABLE AFFECTER
   (
    BAGNUM NUMBER  NOT NULL,
    CONTNUM NUMBER  NOT NULL,
    DATE_H_AFF DATE  NULL
,   CONSTRAINT PK_AFFECTER PRIMARY KEY (BAGNUM, CONTNUM)
   ) ;

-- ----------------------------------------------------------------------------
--       INDEX DE LA TABLE AFFECTER
-- ----------------------------------------------------------------------------

CREATE  INDEX I_FK_AFFECTER_BAGAGE
     ON AFFECTER (BAGNUM ASC)
    ;

CREATE  INDEX I_FK_AFFECTER_CONTAINER
     ON AFFECTER (CONTNUM ASC)
    ;
```

2. Création des références de table

```
ALTER TABLE CONTAINER_VIRTUEL ADD (
     CONSTRAINT FK_CONT_REEL_VIRTUEL
          FOREIGN KEY (CONTREEL)
               REFERENCES CONTAINER_REEL (AERONUM))   ;


ALTER TABLE CONTAINER_REEL ADD (
     CONSTRAINT FK_CONTAINER_REEL_AERO
          FOREIGN KEY (AERONUM)
               REFERENCES AEROPORT (AERONUM))   ;

ALTER TABLE COUPON_VOL ADD (
     CONSTRAINT FK_COUPON_VOL_OCCURRENCE_VOL
          FOREIGN KEY (OCCNUM)
               REFERENCES OCCURRENCE_VOL (OCCNUM))   ;

ALTER TABLE COUPON_VOL ADD (
     CONSTRAINT FK_COUPON_VOL_BILLET
          FOREIGN KEY (BILLNUM)
               REFERENCES BILLET (BILLNUM))   ;

ALTER TABLE RECETTE ADD (
     CONSTRAINT FK_RECETTE_BILLET
          FOREIGN KEY (BILLNUM)
               REFERENCES BILLET (BILLNUM))   ;

ALTER TABLE VOL ADD (
     CONSTRAINT FK_VOL_AEROPORT
          FOREIGN KEY (AERONUM_ARRIVEE)
               REFERENCES AEROPORT (AERONUM))   ;

ALTER TABLE VOL ADD (
     CONSTRAINT FK_VOL_AEROPORT1
          FOREIGN KEY (AERONUM_DEPART)
               REFERENCES AEROPORT (AERONUM))   ;

ALTER TABLE BILLET ADD (
     CONSTRAINT FK_BILLET_TRAJET
          FOREIGN KEY (TRANUM)
               REFERENCES TRAJET (TRANUM))   ;

ALTER TABLE BILLET ADD (
     CONSTRAINT FK_BILLET_CLIENT
          FOREIGN KEY (CLINUM)
               REFERENCES CLIENT (CLINUM))   ;

ALTER TABLE TRAJET ADD (
     CONSTRAINT FK_TRAJET_AEROPORT
          FOREIGN KEY (AERONUM_DEPART)
               REFERENCES AEROPORT (AERONUM))   ;

ALTER TABLE TRAJET ADD (
     CONSTRAINT FK_TRAJET_AEROPORT1
          FOREIGN KEY (AERONUM_ARRIVEE)
               REFERENCES AEROPORT (AERONUM))   ;

ALTER TABLE OCCURRENCE_VOL ADD (
     CONSTRAINT FK_OCCURRENCE_VOL_VOL
          FOREIGN KEY (VOLNUM)
               REFERENCES VOL (VOLNUM))   ;
```

```
ALTER TABLE OCCURRENCE_VOL ADD (
     CONSTRAINT FK_OCCURRENCE_VOL_AEROPORT
          FOREIGN KEY (AERONUM)
               REFERENCES AEROPORT (AERONUM))    ;

ALTER TABLE CONTAINER_VIRTUEL ADD (
     CONSTRAINT FK_CONTAINER_OCCURRENCE_VOL
          FOREIGN KEY (OCCNUM_DESTINER)
               REFERENCES OCCURRENCE_VOL (OCCNUM))    ;

ALTER TABLE CONTAINER_VIRTUEL ADD (
     CONSTRAINT FK_CONTAINER_OCCURRENCE_VOL1
          FOREIGN KEY (OCCNUM_PROVENIR)
               REFERENCES OCCURRENCE_VOL (OCCNUM))    ;

ALTER TABLE BAGAGE ADD (
     CONSTRAINT FK_BAGAGE_BILLET
          FOREIGN KEY (BILLNUM)
               REFERENCES BILLET (BILLNUM))    ;

ALTER TABLE PROVISION ADD (
     CONSTRAINT FK_PROVISION_BILLET
          FOREIGN KEY (BILLNUM)
               REFERENCES BILLET (BILLNUM))    ;

ALTER TABLE DECHARGER ADD (
     CONSTRAINT FK_DECHARGER_BAGAGE
          FOREIGN KEY (BAGNUM)
               REFERENCES BAGAGE (BAGNUM))    ;

ALTER TABLE DECHARGER ADD (
     CONSTRAINT FK_DECHARGER_CONTAINER
          FOREIGN KEY (CONTNUM)
               REFERENCES CONTAINER_VIRTUEL (CONTNUM))    ;

ALTER TABLE CONSTITUER ADD (
     CONSTRAINT FK_CONSTITUER_TRAJET
          FOREIGN KEY (TRANUM)
               REFERENCES TRAJET (TRANUM))    ;

ALTER TABLE CONSTITUER ADD (
     CONSTRAINT FK_CONSTITUER_VOL
          FOREIGN KEY (VOLNUM)
               REFERENCES VOL (VOLNUM))    ;

ALTER TABLE CHARGER ADD (
     CONSTRAINT FK_CHARGER_BAGAGE
          FOREIGN KEY (BAGNUM)
               REFERENCES BAGAGE (BAGNUM))    ;

ALTER TABLE CHARGER ADD (
     CONSTRAINT FK_CHARGER_CONTAINER
          FOREIGN KEY (CONTNUM)
               REFERENCES CONTAINER_VIRTUEL (CONTNUM))    ;

ALTER TABLE AFFECTER ADD (
     CONSTRAINT FK_AFFECTER_BAGAGE
          FOREIGN KEY (BAGNUM)
               REFERENCES BAGAGE (BAGNUM))    ;

ALTER TABLE AFFECTER ADD (
     CONSTRAINT FK_AFFECTER_CONTAINER
          FOREIGN KEY (CONTNUM)
               REFERENCES CONTAINER_VIRTUEL (CONTNUM))    ;
```

3. Préparation des données

```
-- ------------------------------------------------------------------------
--       AEROPORT
-- ------------------------------------------------------------------------
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (33001,'Paris-Charles De Gaulle',11,1000);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (33002,'Toulouse-Blagnac',7,800);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (33003,'Bordeaux-Mérignac',6,800);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (86002,'Urumqi',8,950);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (86001,'Pékin',10,1000);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (10001,'San Francisco',9,900);
Insert into AEROPORT (AERONUM,AERONOM,AEROTAXE,AEROPOIDSMAXCONTAINER) values (43001,'Berlin',9,900);
-- ------------------------------------------------------------------------
--       TRAJET
-- ------------------------------------------------------------------------
Insert into TRAJET values (100001,33001,33003,260,20,10);
Insert into TRAJET values (100002,33003,33001,286.37,20,10);
Insert into TRAJET values (100003,33002,33001,270.31,20,10);
Insert into TRAJET values (100004,33002,33001,260.98,20,10);
Insert into TRAJET values (100005,33001,33002,261.57,20,10);
Insert into TRAJET values (100006,33001,86001,298.37,43,8);
Insert into TRAJET values (100007,33001,86001,309.98,43,8);
Insert into TRAJET values (100008,86001,33001,360.61,43,8);
Insert into TRAJET values (100009,86001,33001,336.05,43,8);
Insert into TRAJET values (100010,86001,10001,735,40,9);
Insert into TRAJET values (100011,86001,86002,99.37,23,10);
Insert into TRAJET values (100012,86001,86002,103.27,23,10);
Insert into TRAJET values (100013,86001,86002,100.35,23,10);
Insert into TRAJET values (100014,86002,86001,102.45,23,10);
Insert into TRAJET values (100015,86002,86001,120.3,23,10);
Insert into TRAJET values (100016,33002,86002,868.87,43,10);
Insert into TRAJET values (100017,33002,86002,1118.4,43,8);
Insert into TRAJET values (100018,86002,33002,1816.65,43,9);
Insert into TRAJET values (100020,33003,86002,857.66,40,9);
Insert into TRAJET values (100022,33002,86001,627.58,40,9);
Insert into TRAJET values (100023,33002,86001,603.55,40,9);
Insert into TRAJET values (100024,86001,33002,678,40,9);
Insert into TRAJET values (100026,33001,86002,650.37,46,8);
Insert into TRAJET values (100027,33001,86002,648.29,46,8);
Insert into TRAJET values (100028,86002,33001,900,46,8);
Insert into TRAJET values (100029,86002,33001,800,46,8);

-- ------------------------------------------------------------------------
--       CLIENT
-- ------------------------------------------------------------------------
Insert into CLIENT  values (1,'absolon',null,null);
Insert into CLIENT  values (2,'adélaïde',null,null);
Insert into CLIENT  values (3,'adèle',null,null);
Insert into CLIENT  values (4,'adolphe',null,null);
Insert into CLIENT  values (5,'adrien',null,null);
Insert into CLIENT  values (6,'adrienne',null,null);
Insert into CLIENT  values (7,'agnès',null,null);
Insert into CLIENT  values (8,'aimé',null,null);
Insert into CLIENT  values (9,'aimée',null,null);
Insert into CLIENT  values (10,'alain',null,null);
```

```
-- ------------------------------------------------------------------------
--       VOL
-- ------------------------------------------------------------------------
Insert into VOL values (7622,33003,33001,to_date('09:55:00','HH24:MI:SS'),to_date('11:10:00','HH24:MI:SS'),150);
Insert into VOL values (7629,33001,33003,to_date('14:40:00','HH24:MI:SS'),to_date('17:05:00','HH24:MI:SS'),150);
Insert into VOL values (7523,33001,33002,to_date('12:20:00','HH24:MI:SS'),to_date('13:50:00','HH24:MI:SS'),200);
Insert into VOL values (4952,33001,33002,to_date('07:15:00','HH24:MI:SS'),to_date('08:50:00','HH24:MI:SS'),200);
Insert into VOL values (7190,33002,33001,to_date('08:15:00','HH24:MI:SS'),to_date('09:40:00','HH24:MI:SS'),250);
Insert into VOL values (9340,86001,33001,to_date('19:30:00','HH24:MI:SS'),to_date('12:35:00','HH24:MI:SS'),200);
Insert into VOL values (554,86001,33001,to_date('12:25:00','HH24:MI:SS'),to_date('07:00:00','HH24:MI:SS'),200);
Insert into VOL values (7777,33001,86001,to_date('00:20:00','HH24:MI:SS'),to_date('06:40:00','HH24:MI:SS'),3);
Insert into VOL values (3450,33001,86001,to_date('00:05:00','HH24:MI:SS'),to_date('05:30:00','HH24:MI:SS'),150);
Insert into VOL values (1329,10001,86001,to_date('17:25:00','HH24:MI:SS'),to_date('13:05:00','HH24:MI:SS'),170);
Insert into VOL values (7345,86002,86001,to_date('21:45:00','HH24:MI:SS'),to_date('01:40:00','HH24:MI:SS'),140);
Insert into VOL values (1291,86002,86001,to_date('14:55:00','HH24:MI:SS'),to_date('19:10:00','HH24:MI:SS'),150);
Insert into VOL values (5699,86002,86001,to_date('09:15:00','HH24:MI:SS'),to_date('14:55:00','HH24:MI:SS'),150);
Insert into VOL values (6885,86001,86002,to_date('14:45:00','HH24:MI:SS'),to_date('19:50:00','HH24:MI:SS'),200);
Insert into VOL values (5700,86001,86002,to_date('16:20:00','HH24:MI:SS'),to_date('21:00:00','HH24:MI:SS'),150);
Insert into VOL values (5800,33001,86002,to_date('16:20:00','HH24:MI:SS'),to_date('21:00:00','HH24:MI:SS'),3);
```

```
-- ----------------------------------------------------------------------------
--       CONSTITUER
-- ----------------------------------------------------------------------------
Insert into CONSTITUER values (100001,7622,1,0);
Insert into CONSTITUER values (100002,7629,1,0);
Insert into CONSTITUER values (100003,7523,1,0);
Insert into CONSTITUER values (100004,4952,1,0);
Insert into CONSTITUER values (100005,7190,1,0);
Insert into CONSTITUER values (100006,9340,1,0);
Insert into CONSTITUER values (100007,554,1,0);
Insert into CONSTITUER values (100008,7777,1,0);
Insert into CONSTITUER values (100009,3450,1,0);
Insert into CONSTITUER values (100010,1329,1,0);
Insert into CONSTITUER values (100011,7345,1,1);
Insert into CONSTITUER values (100012,1291,1,0);
Insert into CONSTITUER values (100013,5699,1,0);
Insert into CONSTITUER values (100014,6885,1,0);
Insert into CONSTITUER values (100015,5700,1,0);
Insert into CONSTITUER values (100016,1291,3,1);
Insert into CONSTITUER values (100016,7523,1,0);
Insert into CONSTITUER values (100016,9340,2,1);
Insert into CONSTITUER values (100017,554,2,1);
Insert into CONSTITUER values (100017,4952,1,0);
Insert into CONSTITUER values (100017,5699,3,1);
Insert into CONSTITUER values (100018,6885,1,0);
Insert into CONSTITUER values (100018,7190,3,1);
Insert into CONSTITUER values (100018,7777,2,1);
Insert into CONSTITUER values (100020,1291,3,1);
Insert into CONSTITUER values (100020,7629,1,0);
Insert into CONSTITUER values (100020,9340,2,1);
Insert into CONSTITUER values (100022,7523,1,0);
Insert into CONSTITUER values (100022,9340,2,1);
Insert into CONSTITUER values (100023,554,2,1);
Insert into CONSTITUER values (100023,4952,1,0);
Insert into CONSTITUER values (100024,7190,2,0);
Insert into CONSTITUER values (100024,7777,1,0);
Insert into CONSTITUER values (100026,1291,2,1);
Insert into CONSTITUER values (100026,9340,1,1);
Insert into CONSTITUER values (100027,554,1,1);
Insert into CONSTITUER values (100027,5699,2,1);
Insert into CONSTITUER values (100028,3450,2,1);
Insert into CONSTITUER values (100028,5700,1,0);
Insert into CONSTITUER values (100029,5800,1,0);
```

```
-- ----------------------------------------------------------------------------
--       CONSTITUER
-- ----------------------------------------------------------------------------
Insert into CONSTITUER values (100001,7622,1,0);
Insert into CONSTITUER values (100002,7629,1,0);
Insert into CONSTITUER values (100003,7523,1,0);
Insert into CONSTITUER values (100004,4952,1,0);
Insert into CONSTITUER values (100005,7190,1,0);
Insert into CONSTITUER values (100006,9340,1,0);
Insert into CONSTITUER values (100007,554,1,0);
Insert into CONSTITUER values (100008,7777,1,0);
Insert into CONSTITUER values (100009,3450,1,0);
Insert into CONSTITUER values (100010,1329,1,0);
Insert into CONSTITUER values (100011,7345,1,1);
Insert into CONSTITUER values (100012,1291,1,0);
Insert into CONSTITUER values (100013,5699,1,0);
Insert into CONSTITUER values (100014,6885,1,0);
Insert into CONSTITUER values (100015,5700,1,0);
Insert into CONSTITUER values (100016,1291,3,1);
Insert into CONSTITUER values (100016,7523,1,0);
Insert into CONSTITUER values (100016,9340,2,1);
Insert into CONSTITUER values (100017,554,2,1);
Insert into CONSTITUER values (100017,4952,1,0);
Insert into CONSTITUER values (100017,5699,3,1);
Insert into CONSTITUER values (100018,6885,1,0);
Insert into CONSTITUER values (100018,7190,3,1);
Insert into CONSTITUER values (100018,7777,2,1);
Insert into CONSTITUER values (100020,1291,3,1);
Insert into CONSTITUER values (100020,7629,1,0);
Insert into CONSTITUER values (100020,9340,2,1);
Insert into CONSTITUER values (100022,7523,1,0);
Insert into CONSTITUER values (100022,9340,2,1);
Insert into CONSTITUER values (100023,554,2,1);
Insert into CONSTITUER values (100023,4952,1,0);
Insert into CONSTITUER values (100024,7190,2,0);
Insert into CONSTITUER values (100024,7777,1,0);
Insert into CONSTITUER values (100026,1291,2,1);
Insert into CONSTITUER values (100026,9340,1,1);
Insert into CONSTITUER values (100027,554,1,1);
Insert into CONSTITUER values (100027,5699,2,1);
Insert into CONSTITUER values (100028,3450,2,1);
Insert into CONSTITUER values (100028,5700,1,0);
Insert into CONSTITUER values (100029,5800,1,0);
```

```
-- ------------------------------------------------------------------------
--       OCCURRENCE_VOL
-- ------------------------------------------------------------------------
Insert into OCCURRENCE_VOL values (580004,5800,33001,sysdate+3,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (580003,5800,33001,sysdate+2,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (580002,5800,33001,sysdate+1,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (762901,7629,33002,sysdate,null);
Insert into OCCURRENCE_VOL values (752301,7523,33003,sysdate,null);
Insert into OCCURRENCE_VOL values (495201,4952,33003,sysdate,null);
Insert into OCCURRENCE_VOL values (719001,7190,33003,sysdate,null);
Insert into OCCURRENCE_VOL values (934001,9340,86002,sysdate,null);
Insert into OCCURRENCE_VOL values (55401,554,86002,sysdate,null);
Insert into OCCURRENCE_VOL values (777701,7777,86002,sysdate,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (345001,3450,86002,sysdate,null);
Insert into OCCURRENCE_VOL values (132901,1329,86002,sysdate,null);
Insert into OCCURRENCE_VOL values (734501,7345,33001,sysdate,null);
Insert into OCCURRENCE_VOL values (129101,1291,33001,sysdate,null);
Insert into OCCURRENCE_VOL values (569901,5699,33001,sysdate,null);
Insert into OCCURRENCE_VOL values (688501,6885,33001,sysdate,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (570001,5700,33001,sysdate,null);
Insert into OCCURRENCE_VOL values (762202,7622,33002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (762902,7629,33002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (752302,7523,33003,sysdate+1,null);
Insert into OCCURRENCE_VOL values (495202,4952,33003,sysdate+1,null);
Insert into OCCURRENCE_VOL values (719002,7190,33003,sysdate+1,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (934002,9340,86002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (55402,554,86002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (777702,7777,86002,sysdate+1,'ouverte à la réservation');
Insert into OCCURRENCE_VOL values (345002,3450,86002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (132902,1329,86002,sysdate+1,null);
Insert into OCCURRENCE_VOL values (734502,7345,33001,sysdate+1,null);
Insert into OCCURRENCE_VOL values (129102,1291,33001,sysdate+1,null);
Insert into OCCURRENCE_VOL values (569902,5699,33001,sysdate+1,null);
Insert into OCCURRENCE_VOL values (688502,6885,33001,sysdate+1,null);
Insert into OCCURRENCE_VOL values (570002,5700,33001,sysdate+1,null);
Insert into OCCURRENCE_VOL values (762201,7622,33002,sysdate,null);

-- ------------------------------------------------------------------------
--        Container_reel
-- ------------------------------------------------------------------------
Insert into CONTAINER_REEL values(1,86002,'oui');
Insert into CONTAINER_REEL values(2,86002,'oui');
Insert into CONTAINER_REEL values(3,86002,'oui');
Insert into CONTAINER_REEL values(4,86002,'oui');
Insert into CONTAINER_REEL values(5,86002,'oui');
Insert into CONTAINER_REEL values(6,86001,'oui');
Insert into CONTAINER_REEL values(7,86001,'oui');
Insert into CONTAINER_REEL values(8,86001,'oui');
Insert into CONTAINER_REEL values(9,86001,'oui');
Insert into CONTAINER_REEL values(10,86001,'oui');
Insert into CONTAINER_REEL values(11,33001,'oui');
Insert into CONTAINER_REEL values(12,33001,'oui');
Insert into CONTAINER_REEL values(13,33001,'oui');
Insert into CONTAINER_REEL values(14,33002,'oui');
Insert into CONTAINER_REEL values(15,33002,'oui');
Insert into CONTAINER_REEL values(16,33002,'oui');
Insert into CONTAINER_REEL values(17,33002,'oui');
Insert into CONTAINER_REEL values(18,43001,'oui');
Insert into CONTAINER_REEL values(19,43001,'oui');
Insert into CONTAINER_REEL values(20,43001,'oui');
Insert into CONTAINER_REEL values(21,10001,'oui');
Insert into CONTAINER_REEL values(22,10001,'oui');
Insert into CONTAINER_REEL values(23,10001,'oui');
Insert into CONTAINER_REEL values(24,10001,'oui');
```

4. Création des séquences

```
DROP SEQUENCE SEQ_BAGAGE;
CREATE SEQUENCE SEQ_BAGAGE INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;

DROP SEQUENCE SEQ_BILLET;
CREATE SEQUENCE SEQ_BILLET INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;

DROP SEQUENCE SEQ_CONTAINER;
CREATE SEQUENCE SEQ_CONTAINER INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;

DROP SEQUENCE SEQ_COUPONVOL;
CREATE SEQUENCE SEQ_COUPONVOL INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;

DROP SEQUENCE SEQ_PROVISION;
CREATE SEQUENCE SEQ_PROVISION INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;

DROP SEQUENCE SEQ_RECETTE;
CREATE SEQUENCE SEQ_RECETTE INCREMENT BY 1 START WITH 0 MAXVALUE 9999999999999999999999999999 MINVALUE 0 CACHE 20;
```

## Partie II : requêtes SQL

### 1. Gestion de billets

1. Vérifier si un client a déjà acheté un billet d'un trajet pour une date de départ.

Il faut vérifier si un client a déjà acheté un billet pour le même trajet et la même date de départ. Si c'est le cas, la procédure d'achat s'arrête parce qu'un client ne peut pas prendre deux places dans une occurrence de vol.

Requête en SQL : Est-ce que le client 1 a déjà acheté un billet du trajet 1 pour le 19/01/2017.

```
SELECT COUNT(*) AS DEJAACHETE
    FROM BILLET
    WHERE CLINUM = 1
    AND TRANUM = 1 ;
```

Résultat d'exécution :

```
DEJAACHETE
----------
         0
```

### 2. Gestion de bagage

1. Trouver la première occurrence de vol d'un billet

Quand un bagage est enregistré par un client, il faut savoir la première occurrence de vol de son billet pour affecter son bagage. Trouver la première occurrence de vol d'un billet

Requête en SQL : Trouver la première occurrence de vol du billet 2

```
SELECT OCC.OCCNUM, OCC.OCCETAT
  FROM CONSTITUER C, OCCURRENCE_VOL OCC, BILLET BIL
  WHERE BIL.TRANUM = C.TRANUM
  AND C.NUMORDRE = 1
  AND C.VOLNUM = OCC.VOLNUM
  AND TO_DATE(OCC.OCCDATE,'DD/MM/YYYY') = TO_DATE(BIL.BILLDATEDEPART,'DD/MM/YYYY')
  AND BIL.BILLNUM = 2;
```

Résultat d'exécution :

```
    OCCNUM OCCETAT
---------- --------------------------------------------------
    688501 arrivée
```

2. Compter le nombre de bagages affectés à un conteneur virtuel et le nombre de bagages chargés ou déchargés de ce conteneur.

Quand des employés scannent les codes-barres des bagages lors du chargement ou déchargement, pour le premier bagage déposé, l'état de conteneur virtuel passe à « en cours de chargement » ou « en cours de déchargement », lorsque tous les bagages affectés à un conteneur virtuel sont chargés ou déchargés, l'état de ce conteneur passe à « chargé » ou « déchargé ».

Nous devons donc compter le nombre de bagages affectés à un conteneur virtuel et le compare avec le nombre de bagages chargés dans ce conteneur ou le nombre de bagages déchargés.

Requête en SQL :

a. Nombre de bagages affectés au conteneur virtuel 3

```
SELECT COUNT(*) AS nbAffecte
  FROM AFFECTER
  WHERE CONTNUM = 3;
```

Résultat d'exécution :

```
 NBAFFECTE
----------
         3
```

b. Nombre de bagages chargés dans le conteneur virtuel 3

```
SELECT COUNT(*) AS nbCharge
  FROM CHARGER
  WHERE CONTNUM = 3;
```

Résultat d'exécution :

```
  NBCHARGE
----------
         3
```

c. Nombre de bagages déchargés du conteneur virtuel 3

```
SELECT COUNT(*) AS nbDecharge
  FROM DECHARGER
  WHERE CONTNUM = 3;
```

Résultat d'exécution :

```
 NBDECHARGE
----------
         3
```

3. Trouver le numéro d'aéroport et sa limite de poids d'un conteneur selon le numéro de l'occurrence provenir et le numéro de l'occurrence destiner.

Quand le système affecte un bagage à un conteneur virtuel, il faut savoir c'est dans lequel aéroport et sa limite de poids d'un conteneur. L'aéroport est l'aéroport d'arrivée de l'occurrence provenir et/ou l'aéroport de départ de l'occurrence destiner.

Requête en SQL : un client a un billet du trajet 100001 qui comprend une occurrence 762201, il doit s'enregistrer à quel aéroport ? et c'est quoi le poids maximum d'un conteneur dans cet aéroport ?

```
SELECT AE.AEROPOIDSMAXCONTAINER, AE.AERONUM
  FROM AEROPORT AE, OCCURRENCE_VOL OCC, VOL V
  WHERE AE.AERONUM = V.AERONUM_DEPART
  AND V.VOLNUM = OCC.VOLNUM
  AND OCC.OCCNUM = 762201;
```

Résultat d'exécution :

```
                AEROPOIDSMAXCONTAINER    AERONUM
------------------------------------- ----------
                                 1000      33001
```

*Quand nous ne sont pas sûrs pour l'ordre de l'occurrence de vol, nous utilisons une requête plus générale :

```
SELECT AE.AEROPOIDSMAXCONTAINER, AE.AERONUM
  FROM AEROPORT AE, OCCURRENCE_VOL OCC, VOL V
  WHERE AE.AERONUM = V.AERONUM_ARRIVEE
  AND V.VOLNUM = OCC.VOLNUM
  AND OCC.OCCNUM = X --(Numéro de l'occurrence provenir)
UNION
SELECT AE.AEROPOIDSMAXCONTAINER, AE.AERONUM
  FROM AEROPORT AE, OCCURRENCE_VOL OCC, VOL V
  WHERE AE.AERONUM = V.AERONUM_DEPART
  AND V.VOLNUM = OCC.VOLNUM
  AND OCC.OCCNUM = X; --(Numéro de l'occurrence destiner)
```

## 3. Gestion d'occurrence

1. Trouver le coupon de vol pour un client et une occurrence de vol

Quand un client s'enregistre pour une occurrence de vol, le coupon correspondant passe dans « enregistré ».

Requête en SQL : Trouver le numéro d'occurrence de vol pour le client

```
SELECT CV.COUPNUM
  FROM BILLET B,COUPON_VOL CV
  WHERE B.BILLNUM=CV.BILLNUM
  AND B.CLINUM=1
  AND CV.OCCNUM=719002;
```

Résultat d'exécution :

```
   COUPNUM
----------
         1
```

2. Trouver tous les coupons d'une occurrence de vol qui ne sont pas en état « enregistré ».

Lorsqu'une occurrence de vol passe dans l'état décollé, tous les coupons correspondants qui ne sont pas dans l'état enregistré, passent automatiquement dans l'état annulé. Nous devons donc trouver tous les coupons correspondants pour que nous puissions changer ses états.

Requête en SQL : Trouver les coupons qui ne sont pas en état de « enregistré » de l'occurrence de vol 580004

```
select coupnum
from coupon_vol
where OCCNUM=580004
and COUPETAT<>'enregistré';
```

Résultat d'exécution :

```
   COUPNUM
----------
        10
        11
        12
```

3. Trouver une occurrence de vol si un de ses occurrences de trajet est dérouté à une date

Lorsqu'une occurrence de vol passe dans l'état décollé, tous les coupons correspondants qui ne sont pas dans l'état enregistré, passent automatiquement dans l'état annulé. Nous devons donc trouver tous les coupons correspondants pour que nous puissions changer ses états.

Requête en SQL : Trouver les coupons qui ne sont pas en état de « enregistré » de l'occurrence de vol 580004

```
select coupnum
from coupon_vol
where OCCNUM=580004
and COUPETAT<>'enregistré';
```

Résultat d'exécution :

```
  COUPNUM
----------
        10
        11
        12
```

4. Si une occurrence de vol est déroutée, avec le numéro de nouvelle occurrence de vol, le trajet et la date que le client a choisies il faut trouver la prochaine occurrence de vol.

Lorsqu'une occurrence de vol passe dans l'état dérouté, les clients vont avoir des coupons correspondent à une nouvelle occurrence de vol. Comme cette nouvelle occurrence de vol n'existe pas dans ses trajets originaux, il faut trouver les prochaines occurrences selon ses billets.

Requête en SQL : le client qui a acheté un billet pour le trajet 10018 pour la date de départ 14/01/2018 a pris le vol 5800 à cause de l'occurrence de vol originale été déroutée. Quel est la prochaine occurrence de vol pour ce client ?

```
SELECT OV.OCCNUM
FROM CONSTITUER C,VOL V,OCCURRENCE_VOL OV
WHERE C.TRANUM=100018
AND C.VOLNUM=V.VOLNUM
AND OV.VOLNUM=V.VOLNUM
AND To_date(TO_char(OV.occdate,'dd/mm/yy'),'dd/mm/yy')=TO_DATE('14/01/18','dd/mm/yy')+C.JOURPLUS
AND V.AERONUM_DEPART=(
  SELECT AERONUM_ARRIVEE
  FROM VOL V2
  WHERE V2.VOLNUM=5800
);
```

Résultat d'exécution :

```
  OCCNUM
----------
    719002
```

5. Si une occurrence de vol est déroutée, trouver la première occurrence de vol dans laquelle il reste des places et qui part de la destination sur laquelle a été dérouté l'avion pour arriver à la destination finale du vol dérouté

La nouvelle occurrence de vol doit avoir la même destination que l'occurrence de vol déroutée, et c'est la première occurrence de vol qui a des sièges disponibles.

Requête : Si l'occurrence de vol 777702 est déroutée, pour un des clients qui ont besoins des nouvelles occurrences de vol, il peut prendre laquelle occurrence ?

```
SELECT OV.OCCNUM
FROM  VOL V,OCCURRENCE_VOL OV
LEFT JOIN COUPON_VOL CV ON OV.OCCNUM=CV.OCCNUM
WHERE V.AERONUM_DEPART=(
    SELECT AERONUM
    FROM OCCURRENCE_VOL OV1
    WHERE OV1.OCCNUM=777702
)
AND V.AERONUM_ARRIVEE=(
    SELECT AERONUM_ARRIVEE
    FROM VOL V2,OCCURRENCE_VOL OV2
    WHERE V2.VOLNUM=OV2.VOLNUM
    AND OV2.OCCNUM=777702
)
AND OV.VOLNUM=V.VOLNUM
AND TO_DATE(TO_CHAR(OV.OCCDATE,'dd/mm/yy')||'-'||TO_CHAR(V.H_DEPART,'HH24:MI:SS'),'dd/mm/yy-HH24:MI:SS')>SYSDATE
AND OV.OCCETAT='ouverte à la réservation'
AND ROWNUM=1
GROUP BY OV.OCCNUM,V.VOLNBPLACES,OCCDATE
HAVING COUNT(CV.COUPNUM)<V.VOLNBPLACES
ORDER BY OCCDATE ASC;
```

Résultat d'exécution :

```
    OCCNUM
----------
    580003
```

## 4. Gestion financière

1. Trouver la limite de poids de bagage et le prix supplémentaire pour un kilo d'un trajet.

A l'enregistrement des bagages, on doit créditer sur le compte de provision le montant pour la surcharge des bagages de ce client.

Requête en SQL : Trouver la limite de poids de bagage et le prix supplémentaire d'un kilo pour le trajet 1.

```
SELECT T.TRANKGBAG, T.TRATARIFKGSUP
  FROM TRAJET T, BILLET BIL
  WHERE BIL.TRANUM = T.TRANUM
  AND BIL.BILLNUM = 1;
```

Résultat d'exécution :

```
 TRANKGBAG                          TRATARIFKGSUP
---------- ----------------------------------------
        43                                        9
```

## 5. Gestion de conteneurs

1. Trouver un conteneur disponible dans un aéroport.

Les bagages enregistrés sont d'abord affectés à des conteneurs virtuels, lorsqu'un conteneur virtuel est créé, il faut le lier avec un conteneur disponible dans cet aéroport.

Requête en SQL : trouver un conteneur disponible dans l'aéroport

```
SELECT CONTNUM
FROM   CONTAINER_REEL
WHERE  AERONUM = 86002
AND ROWNUM = 1
 AND CONTNUM NOT IN(SELECT CONTREEL
                    FROM   CONTAINER_VIRTUEL);
```

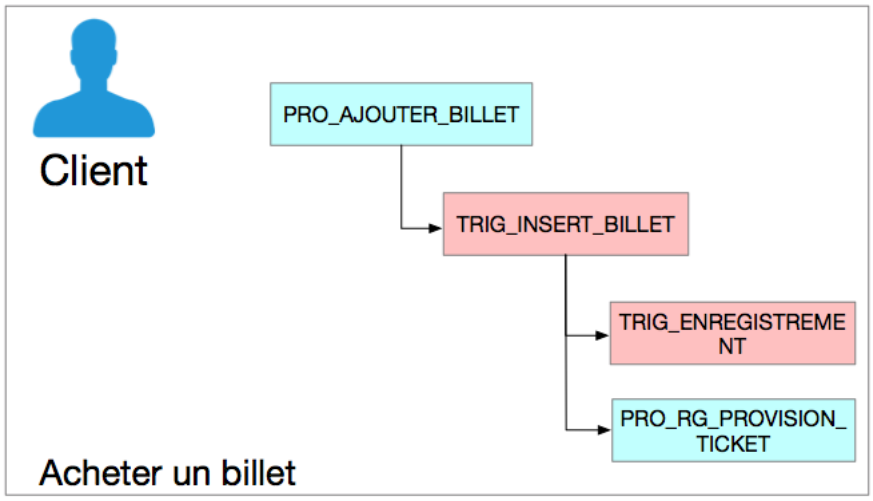
Résultat d'exécution :

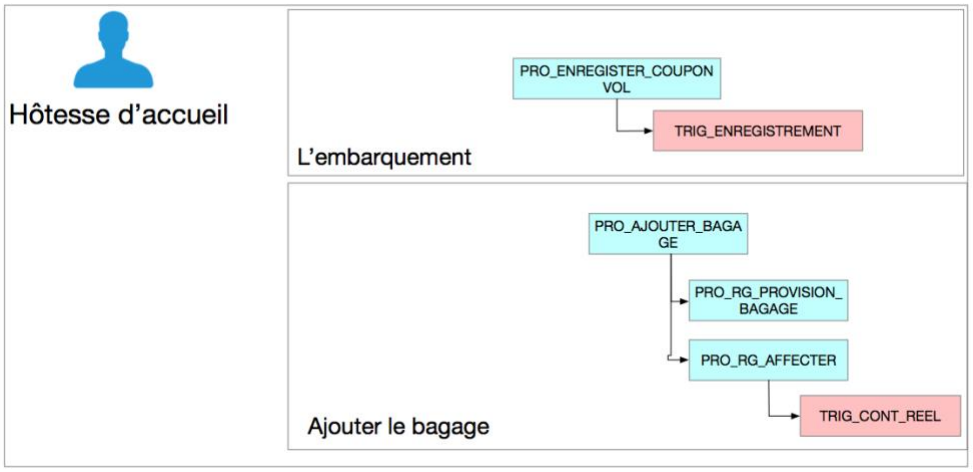
```
   CONTNUM
----------
         3
```

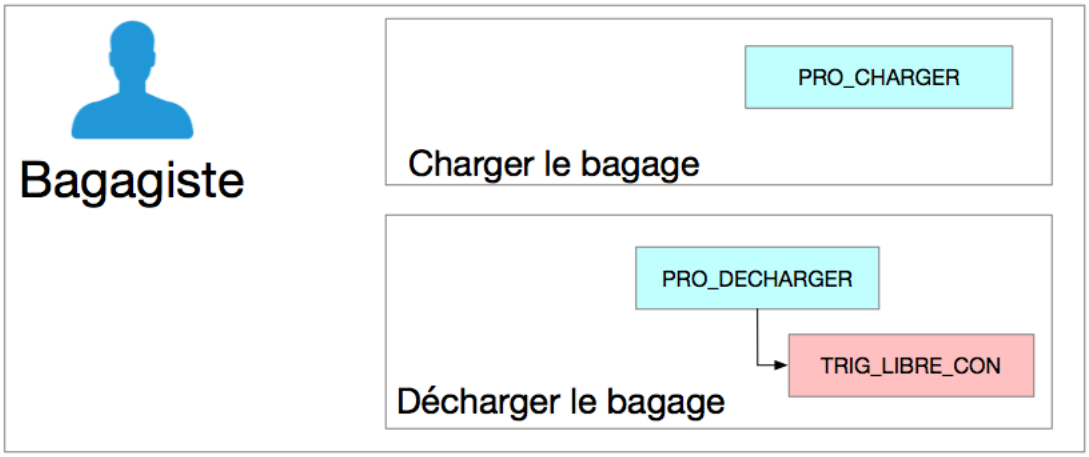
## Partie III : programmation PL/SQL

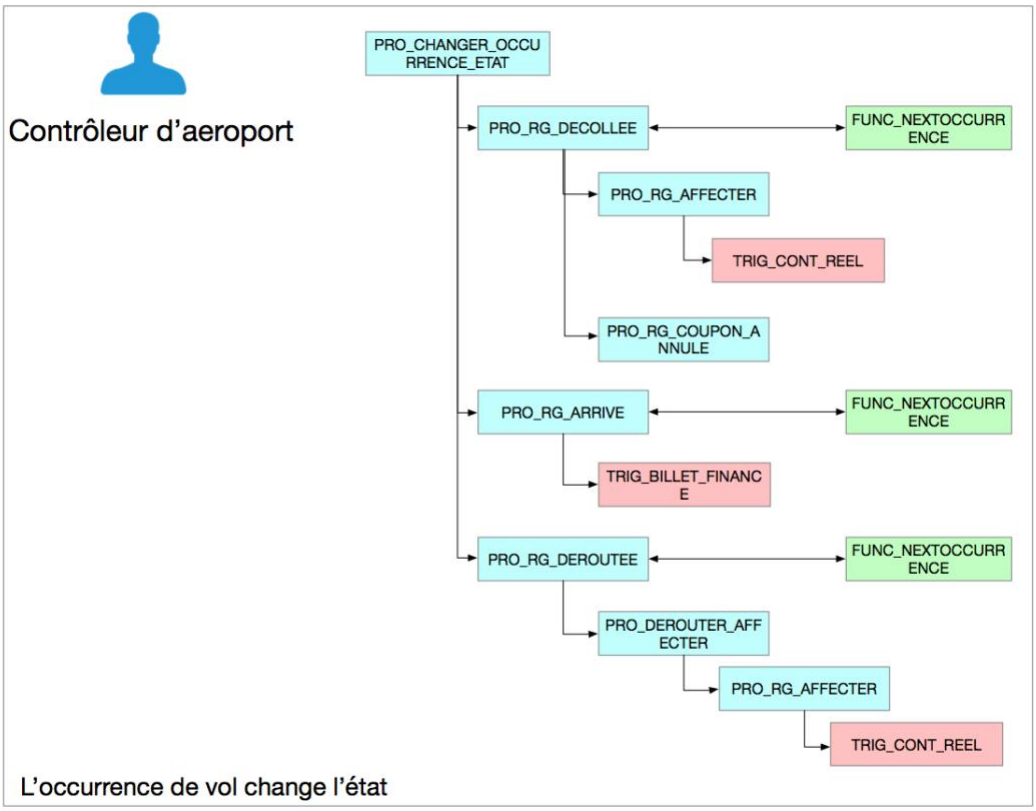
### 1. Schéma général

Afin de simplifier et pratiquer l'utilisation de notre produit, on définit 4 types d'utilisateur et 6 opérations (API) correspondants. Les schémas suivants montrent les relations entre l'utilisateur et l'opération et entre les processus internes.






## 2. Gestion de billets

### *2.1. PRO_AJOUTER_BILLET*

Pour acheter un billet, le client doit saisir son numéro, le numéro du trajet choisi et une date de départ. La procédure d'abord vérifie si ce client a déjà acheté un billet pour le même trajet et la même date de départ. Si c'est le cas, la procédure d'achat s'arrête parce qu'un client ne peut pas prendre 2 places dans une occurrence de vol. Si non, le système insère un billet dans la table « BILLET » et affiche le résultat d'achat.

```
create or replace PROCEDURE PRO_AJOUTER_BILLET (
    P_TRAJETNUM     IN TRAJET.TRANUM%TYPE,
    P_DATEDEPART    IN BILLET.BILLDATEDEPART%TYPE,
    P_CLIENTNUM     IN CLIENT.CLINUM%TYPE
) AS
    V_RESULTAT_CLI          NUMBER;
    V_AFFICH_AERODEPART     AEROPORT.AERONOM%TYPE;
    V_AFFICH_AEROARRIVEE    AEROPORT.AERONOM%TYPE;
BEGIN
    --Vérifier si ce client a déjà acheté le même billet pour le même trajet da la même date de départ
    SELECT COUNT(*) INTO V_RESULTAT_CLI
        FROM BILLET
        WHERE CLINUM = P_CLIENTNUM
        AND TRANUM = P_TRAJETNUM
        AND BILLDATEDEPART = P_DATEDEPART;
    --Si v_resultat_cli>0, il signifie que ce client a déjà acheté ce trajet pour la date de départ
    IF V_RESULTAT_CLI > 0 THEN
        RAISE_APPLICATION_ERROR(-20000,
            'Le trajet '|| P_TRAJETNUM || ' ont été déjà acheté par client :'|| P_CLIENTNUM|| ' pour la date:'|| P_DATEDEPART);
    END IF;
    --Inserer les donées dans la table billet et ça va déclencher un trigger pour vérifier la disponibilité de ce billet
    INSERT INTO BILLET VALUES (SEQ_BILLET.NEXTVAL, P_TRAJETNUM, P_CLIENTNUM, SYSDATE, P_DATEDEPART, 'émis');
    --AFFICHER LE RÉSULTAT
    SELECT A1.AERONOM,A2.AERONOM INTO V_AFFICH_AERODEPART,V_AFFICH_AEROARRIVEE
        FROM TRAJET T,AEROPORT A1,AEROPORT A2
        WHERE A1.AERONUM = T.AERONUM_DEPART
        AND   A2.AERONUM = T.AERONUM_ARRIVEE
        AND   T.TRANUM = P_TRAJETNUM ;
    DBMS_OUTPUT.PUT_LINE('Client '|| P_CLIENTNUM||' a acheté le billet '|| SEQ_BILLET.CURRVAL||' pour le trajet '
                         || P_TRAJETNUM || ' de '||V_AFFICH_AERODEPART||' à '||V_AFFICH_AEROARRIVEE);
    COMMIT;
END PRO_AJOUTER_BILLET;
```

### 2.2. PRO_ENREGISTER_COUPONVOL

Cette procédure est pour que des clients peuvent s'enregistrent ses coupons.

D'abord, nous vérifie si le client a bien ce coupon, si oui, l'état de ce coupon passe à « enregistré » ; si non, la procédure s'arrête.

```
create or replace PROCEDURE PRO_ENREGISTER_COUPONVOL
(
  P_CLINUM IN CLIENT.CLINUM%TYPE
, P_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
V_COUPNUM COUPON_VOL.COUPNUM%TYPE;
BEGIN
    --TROUVER LE COUPON DE VOL PAR CLINUM ET OCCNUM
    SELECT CV.COUPNUM INTO V_COUPNUM
        FROM BILLET B,COUPON_VOL CV
        WHERE B.BILLNUM=CV.BILLNUM
        AND B.CLINUM=P_CLINUM
        AND CV.OCCNUM=P_OCCNUM;
    --Mis en jour à l'état de coupon de vol
    UPDATE COUPON_VOL SET COUPETAT='enregistré' WHERE COUPNUM=V_COUPNUM;
    --Afficher le résultat
    SYS.DBMS_OUTPUT.PUT_LINE('Le client '||P_CLINUM ||' a réussi de enregistre son coupon de vol '||P_OCCNUM);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20006,
            'Le client '||P_CLINUM||' n''a aucun coupon de vol '||P_OCCNUM);
END PRO_ENREGISTER_COUPONVOL;
```

### 2.3. TRIG_ENREGISTREMENT

Avant de mettre à jour l'état d'un coupon, nous vérifions si l'occurrence correspondante est en état de « ouverte à l'embarquement ». Si non, le coupon ne peut pas être mettre à jour ; si oui, nous regardons si le coupon est le premier coupon du billet correspondant, si c'est le cas, l'état de ce billet passe à « en cours ».

```
create or replace TRIGGER TRIG_ENSEIGNEMENT
BEFORE UPDATE OF COUPETAT ON COUPON_VOL
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
WHEN (N.coupETAT='enregistré')
DECLARE
v_ouverte_embarquement VARCHAR2(50);
v_fisrt_occnum number;
BEGIN
 select occetat into v_ouverte_embarquement from OCCURRENCE_VOL where OCCNUM=:N.occnum;
 IF v_ouverte_embarquement <> 'ouverte à l''embarquement' THEN
  RAISE_APPLICATION_ERROR(-20100,'L''occurrence de vol '||:N.occnum||' n''est pas ouvete à l''embarquement');
  end if;

  SELECT OCC.OCCNUM into v_fisrt_occnum
FROM CONSTITUER C, OCCURRENCE_VOL OCC, BILLET BIL
WHERE BIL.TRANUM = C.TRANUM
AND C.NUMORDRE = 1
AND C.VOLNUM = OCC.VOLNUM
AND TO_DATE(OCC.OCCDATE,'DD/MM/YYYY') = TO_DATE(BIL.BILLDATEDEPART,'DD/MM/YYYY')
AND BIL.BILLNUM = :N.BILLNUM;

if v_fisrt_occnum=:N.occnum then
update BILLET SET BILLETAT='en cours' where BILLNUM=:N.billnum;
END IF;
END;
```

### 2.4. TRIG_INSERT_BILLET

Si un client veut acheter un billet d'un trajet pour une date de départ. D'abord, nous trouvons toutes les occurrences de vol correspondants à ce trajet. S'il n'y a pas d'occurrence correspondant, l'achat est échoué. S'il y a des occurrences de vol trouvées, pour chaque occurrence de vol nous regardons si c'est en état « ouverte à la réservation » et s'il y a encore des sièges disponibles. Si oui, l'achat est réussi et nous créons des coupons correspondants automatiquement.

```
create or replace TRIGGER TRIG_INSERT_BILLET
AFTER INSERT ON "DBA_PROJET_BESTGROUP"."BILLET"
REFERENCING OLD AS "O" NEW AS "N"
FOR EACH ROW
DECLARE

CURSOR cur_occurences IS
  select O.occnum,OccEtat,VolNbPlaces,count(CV.coupnum) as NbPlaceAchetee
  from constituer C, vol V,OCCURRENCE_VOL O
  left join COUPON_VOL CV
  on O.OCCNUM=CV.OCCNUM
  where :N.TRANUM=C.TRANUM
  and C.VOLNUM=V.VOLNUM
  and O.volnum=V.VOLNUM
  and TO_DATE(O.occdate,'dd/mm/yy')=TO_DATE(:N.billdatedepart+C.JOURPLUS,'dd/mm/yy')
  group by O.occnum,OccEtat,VolNbPlaces;
BEGIN
    for occRecord in cur_occurences loop
    IF occRecord.OccEtat is null THEN
        RAISE_APPLICATION_ERROR(-20001, 'La trajet '||:N.tranum||
        ' n''ont aucun occurrence de vol pour la date de départ '||:N.billdatedepart);
    ELSIF occRecord.OccEtat<>'ouverte à la réservation' THEN
        RAISE_APPLICATION_ERROR(-20002, 'Les occurrences de vol de la trajet '||:N.tranum
        ||' ne sont pas encore ouvertées à la réservation pour la date de départ '||:N.billdatedepart);
    ELSIF occRecord.NbPlaceAchetee=occRecord.VolNbPlaces then
        RAISE_APPLICATION_ERROR(-20003, 'Les occurrences de vol de la trajet '||:N.tranum
        ||' n''ont pas assez de place libre pour la date de départ '||:N.billdatedepart);
    END IF;
   END LOOP;

  for occRecord in cur_occurences loop
      insert into coupon_vol values(SEQ_COUPONVOL.nextVal,occRecord.occnum,:N.billnum,'réservé');
  END LOOP;

  PRO_RG_PROVISION_TICKET(:N.billnum,:N.tranum);
END;
```

## 3. Gestion d'occurrences

### *3.1. PRO_CHANGER_OCCURRENCE_ETAT*

Avant de changer l'état d'une occurrence, nous utilise cette procédure pour vérifier la possibilité de changement.

D'abord, nous trouvons l'état actuel de l'occurrence. Ensuite, on compare l'état actuel avec le nouvel état.

- Si le nouvel état est « ouverte à l'embarquement », nous vérifions si l'état actuel est « ouverte à la réservation ».
- Si le nouvel état est « décollée », nous vérifions si l'état actuel est « ouverte à l'embarquement ».
- Si le nouvel état est « arrivée», nous vérifions si l'état actuel est « décollée ».
- Si le nouvel état est « déroutée», nous vérifions si l'état actuel est « décollée».

Si les états ne sont pas correspondants, la procédure s'arrête.

```
create or replace PROCEDURE PRO_CHANGER_OCCURRENCE_ETAT
(
    P_OCCNUM  IN OCCURRENCE_VOL.OCCNUM%TYPE
,   P_ETAT    IN OCCURRENCE_VOL.OCCETAT%TYPE
) AS
    V_ETAT_AVANT OCCURRENCE_VOL.OCCETAT%TYPE;
BEGIN
  --On trouve l'état d'occurrence précédent.
    SELECT OCCETAT  INTO V_ETAT_AVANT FROM OCCURRENCE_VOL WHERE OCCNUM=P_OCCNUM;
  --Executer le processus selon l'état à changer
    CASE P_ETAT
    WHEN 'ouverte à l''embarquement' THEN
        IF V_ETAT_AVANT<>'ouverte à la réservation' THEN
            RAISE_APPLICATION_ERROR(-20301,
                'L''état d''occurrence de vol '||P_OCCNUM||' devrait être <ouverte à la réservation>');
        END IF;
    WHEN 'décollée' THEN
        IF V_ETAT_AVANT='ouverte à l''embarquement' THEN
            PRO_RG_DECOLLEE(P_OCCNUM);
        ELSE
            RAISE_APPLICATION_ERROR(-20301,
                'L''état d''occurrence de vol '||P_OCCNUM||' devrait être <ouverte à l''embarquement>');
        END IF;
    WHEN 'arrivée' THEN
        IF V_ETAT_AVANT='décollée' THEN
            PRO_RG_ARRIVE(P_OCCNUM);
        ELSE
            RAISE_APPLICATION_ERROR(-20302,'L''état d''occurrence de vol '||P_OCCNUM||' devrait être <décollée>');
        END IF;
    WHEN 'déroutée' THEN
        IF V_ETAT_AVANT='décollée' THEN
            PRO_RG_DECOUTEE(P_OCCNUM);
        ELSE
            RAISE_APPLICATION_ERROR(-20302,'L''état d''occurrence de vol '||P_OCCNUM||' devrait être <décollée>');
        END IF;
    ELSE NULL;
    END CASE;
    --Après les processus correspondants fini, on met en jour l'état de l'occurrence de vol et affiche le resultat
    UPDATE OCCURRENCE_VOL SET OCCETAT=P_ETAT WHERE OCCNUM=P_OCCNUM;
    DBMS_OUTPUT.PUT_LINE('L''occurrence de vol '||P_OCCNUM||' est passée dans l''état '||P_ETAT);
END PRO_CHANGER_OCCURRENCE_ETAT;
```

### *3.2. PRO_RG_DECOLLEE*

Dans cette procédure, nous utilisons un curseur pour une liste de tous les billets correspondants, un autre curseur pour tous les bagages correspondants. Pour chaque billet, nous comparons le vol de l'occurrence constitue ce billet, si non, c'est-à-dire cette occurrence de vol est une nouvelle occurrence de vol à cause d'un déroutage.

En cas de déroutage, nous trouvons les nouvelles occurrences de vol pour réaffecter les bagages. S'il n'y a pas de déroutage, nous affectons les bagages selon les trajets originaux.

```
create or replace PROCEDURE PRO_RG_DECOLLEE
(
  P_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
--Tous les billet correspondants
CURSOR CUR_LIST IS
SELECT CV.BILLNUM,B.BILLDATEDEPART
    FROM COUPON_VOL CV,BILLET B
    WHERE B.BILLNUM=CV.BILLNUM
    AND CV.OCCNUM=P_OCCNUM;
--Tous les bagages correspondants
CURSOR CUR_BAGAGES(V_BILLNUM BILLET.BILLNUM%TYPE) IS
    SELECT DISTINCT(BAGNUM) AS BAGNUM
    FROM BAGAGE
    WHERE BILLNUM=V_BILLNUM;
V_OCCNUM_DESTINER OCCURRENCE_VOL.OCCNUM%TYPE ;
V_VOLNUM VOL.VOLNUM%TYPE;
V_TRAJET TRAJET.TRANUM%TYPE;
V_FALG_DEROUTER NUMBER;
V_INPUT_BILLNUM BILLET.BILLNUM%TYPE;
BEGIN
    FOR V_LIST IN CUR_LIST LOOP
        --Trouver le numéro de vol de cette orrucrrence
        SELECT VOLNUM INTO V_VOLNUM FROM OCCURRENCE_VOL WHERE OCCNUM=P_OCCNUM;
        --Trouver le numéro de billet
        SELECT TRANUM INTO V_TRAJET FROM BILLET WHERE BILLNUM=V_LIST.BILLNUM;
        /*Si une occurrence de vol est déroutée, avec le numéro de nouvelle occurrence de vol,
        le trajet et la date que le client a choisies il faut trouver la prochaine occurrence de vol.
        Lorsqu¿une occurrence de vol passe dans l¿état dérouté, les clients vont avoir des coupons correspondent à une nouvelle occurrence de vol.
        Comme cette nouvelle occurrence de vol n¿existe pas dans ses trajets originaux,
        il faut trouver les prochaines occurrences selon ses billets.*/
        SELECT COUNT(*) INTO V_FALG_DEROUTER FROM CONSTITUER WHERE TRANUM=V_TRAJET AND VOLNUM=V_VOLNUM;
        IF V_FALG_DEROUTER=1 THEN
            --CETTE OCCURRENCE EST DANS LE TRAJET
            V_OCCNUM_DESTINER:=FUNC_NEXTOCCURRENCE(P_OCCNUM,V_LIST.BILLNUM);
        ELSE
            --CETTE OCCURRENCE EST DÉROUTÉE
            SELECT OV.OCCNUM INTO V_OCCNUM_DESTINER
                FROM CONSTITUER C,VOL V,OCCURRENCE_VOL OV
                WHERE C.TRANUM=V_TRAJET
                AND C.VOLNUM=V.VOLNUM
                AND OV.VOLNUM=V.VOLNUM
                AND TO_DATE(TO_CHAR(OV.OCCDATE,'dd/mm/yy'),'dd/mm/yy')=TO_DATE(TO_CHAR(V_LIST.BILLDATEDEPART,'dd/mm/yy'),'dd/mm/yy')+C.JOURPLUS
                AND V.AERONUM_DEPART=(
                    SELECT AERONUM_ARRIVEE FROM VOL V2 WHERE V2.VOLNUM=V_VOLNUM
                );
        END IF;
        --Quand l'occurrence de vol décolle, on affect les bagages
        --Trouver les bagages de passenger
        OPEN CUR_BAGAGES(V_LIST.BILLNUM);
        LOOP
            FETCH CUR_BAGAGES INTO V_INPUT_BILLNUM;
            EXIT WHEN CUR_BAGAGES%NOTFOUND;
            PRO_RG_AFFECTER(V_INPUT_BILLNUM,P_OCCNUM,V_OCCNUM_DESTINER);
        END LOOP;
        CLOSE CUR_BAGAGES;
    END LOOP;
    --Si l'état de coupon n'est pas enregistré, on va changer l'état de ces coupons de vol à <anullée>
    PRO_RG_COUPON_ANNULE(P_OCCNUM);

COMMIT;
END PRO_RG_DECOLLEE;
```

### 3.3. PRO_RG_DEROUTEE

Quand une occurrence de vol est déroutée, nous trouvons tous les coupons et tous les bagages correspondants.

Pour chaque coupon, son état passe à « arrivée » et nous trouvons une occurrence de vol qui est la première occurrence de vol dans laquelle il reste des places et qui part de la destination sur laquelle a été dérouté l'avion pour arriver à la destination finale du vol dérouté.

Ensuite, tous les nouveaux coupons passent ses états dans <réservé>.

Pour chaque bagage, nous utilisons la procédure « PRO_DEROUTER_AFFECTER » pour annuler l'ancienne affectation des bagages et réaffecter des bagages.

```
create or replace PROCEDURE PRO_RG_DEROUTEE
(
  P_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
CURSOR CUR_COUPON IS
    SELECT COUPNUM,BILLNUM
    FROM COUPON_VOL CV
    WHERE OCCNUM=P_OCCNUM;
CURSOR CUR_BAGAGES(V_BILLNUM BILLET.BILLNUM%TYPE) IS
    SELECT DISTINCT(BAGNUM) AS BAGNUM
    FROM BAGAGE
    WHERE BILLNUM=V_BILLNUM;
V_NOUVEL_OCCNUM OCCURRENCE_VOL.OCCNUM%TYPE ;
V_OCC_DESTINER_AVANT OCCURRENCE_VOL.OCCNUM%TYPE ;
V_INPUT_BILLNUM BILLET.BILLNUM%TYPE;
BEGIN
    FOR V_COUPON IN CUR_COUPON LOOP
        --Passer les coupons de vol dans l'état <arrivée>
        UPDATE COUPON_VOL SET COUPETAT='arrivée' WHERE COUPNUM=V_COUPON.COUPNUM;
        --Trouver l'occurrence de vol disponible la plus proche
        SELECT OV.OCCNUM INTO V_NOUVEL_OCCNUM
            FROM  VOL V,OCCURRENCE_VOL OV
            LEFT JOIN COUPON_VOL CV ON OV.OCCNUM=CV.OCCNUM
            WHERE V.AERONUM_DEPART=(
                SELECT AERONUM
                    FROM OCCURRENCE_VOL OV1
                    WHERE OV1.OCCNUM=P_OCCNUM)
            AND V.AERONUM_ARRIVEE=(
                SELECT AERONUM_ARRIVEE
                    FROM VOL V2,OCCURRENCE_VOL OV2
                    WHERE V2.VOLNUM=OV2.VOLNUM
                    AND OV2.OCCNUM=P_OCCNUM)
            AND OV.VOLNUM=V.VOLNUM
            --Trouver les occurrences dont la date de départ doit être suivante de le moment où cette processus est executé
            AND TO_DATE(TO_CHAR(OV.OCCDATE,'dd/mm/yy')||'-'||TO_CHAR(V.H_DEPART,'HH24:MI:SS'),'dd/mm/yy-HH24:MI:SS')>SYSDATE
            AND OV.OCCETAT='ouverte à la réservation'
            AND ROWNUM=1    --La première ligne est l'occurrence de vol disponible la plus proche
            GROUP BY OV.OCCNUM,V.VOLNBPLACES,OCCDATE
            HAVING COUNT(CV.COUPNUM)<V.VOLNBPLACES --Trouver les occurrences disponibles dont places est disponible
            ORDER BY OCCDATE ASC; --Mis en ordre les occurrences par la date pour trouver l'occurrence la plus proche
        --On crée les nouvels coupons de vol dont l'état est réservé
        INSERT INTO COUPON_VOL VALUES(SEQ_COUPONVOL.NEXTVAL,V_NOUVEL_OCCNUM,V_COUPON.BILLNUM,'réservé');
        --On affect les bagages
        --On trouve l'occurrence de vol prochaine pour permettre le processus <PRO_DEROUTER_AFFECTER> de trouver et supprimer
        --les containers anciens affectés  qui sont déjà affecté quand cette occurrence décollé.
        V_OCC_DESTINER_AVANT:=FUNC_NEXTOCCURRENCE(P_OCCNUM,V_COUPON.BILLNUM);
        OPEN CUR_BAGAGES(V_COUPON.BILLNUM);
        LOOP
            FETCH CUR_BAGAGES INTO V_INPUT_BILLNUM;
            EXIT WHEN CUR_BAGAGES%NOTFOUND;
            PRO_DEROUTER_AFFECTER(V_INPUT_BILLNUM,V_OCC_DESTINER_AVANT,V_NOUVEL_OCCNUM,P_OCCNUM);
        END LOOP;
        CLOSE CUR_BAGAGES;
    END LOOP;
  COMMIT;
EXCEPTION
 WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR(-20200,'Il n''y pas de vol correspondant');
END PRO_RG_DEROUTEE;
```

### 3.4. PRO_RG_ARRIVE

Quand une occurrence de vol est arrivée, il faut utiliser cette procédure pour mise à jour la base de données.

D'abord nous trouvons tous les coupons correspondant à cette occurrence. Pour chaque coupon, nous changeons son état en « arrivée ». Après, nous regardons que si l'aéroport d'arrivée est la destination du billet correspondant, si c'est le cas, ce coupon est le dernier coupon d'un billet. L'état de ce billet passe en état dans « terminé »

```
create or replace PROCEDURE PRO_RG_ARRIVE
(
  I_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
V_AERONUM_ARRIVEE_TRAJET   AEROPORT.AERONUM%TYPE;
V_AERONUM_ARRIVEE_OCCU     AEROPORT.AERONUM%TYPE;
--Trouver tous les coupons correspondants
CURSOR CUR_COUPON IS
    SELECT COUPNUM,B.BILLNUM,TRANUM
    FROM COUPON_VOL CV,BILLET B
    WHERE CV.OCCNUM=I_OCCNUM
    AND B.BILLNUM=CV.BILLNUM
    AND CV.COUPETAT='enregistré';
V_NEXT_OCCNUM OCCURRENCE_VOL.OCCNUM%TYPE ;
BEGIN
    FOR V_COUPON IN CUR_COUPON LOOP
        --Mis en jour l'état de coupon de vol en <arrivée>
        UPDATE COUPON_VOL SET COUPETAT='arrivée' WHERE COUPNUM=V_COUPON.COUPNUM;
        --Vérifier que si c'est la dernière occurrence de vol dans ce trajet, on mettra l'état de billet en <terminé>
        SELECT AERONUM_ARRIVEE INTO V_AERONUM_ARRIVEE_TRAJET FROM TRAJET WHERE TRANUM=V_COUPON.TRANUM;
        SELECT AERONUM_ARRIVEE INTO V_AERONUM_ARRIVEE_OCCU FROM VOL V,OCCURRENCE_VOL CV WHERE CV.VOLNUM=V.VOLNUM AND CV.OCCNUM=I_OCCNUM;
        V_NEXT_OCCNUM:=FUNC_NEXTOCCURRENCE(I_OCCNUM,V_COUPON.BILLNUM);
        IF V_AERONUM_ARRIVEE_TRAJET=V_AERONUM_ARRIVEE_OCCU THEN
            UPDATE BILLET SET BILLETAT='terminé' WHERE BILLNUM=V_COUPON.BILLNUM;
        END IF;
    END LOOP;
    COMMIT;
END PRO_RG_ARRIVE;
```

### 3.5. PRO_RG_COUPON_ANNULE

Dans ce procédure, nous regardons d'abord tous les coupons qui ne sont pas en état de « enregistré » d'une occurrence de vol. Pour tous les coupons non enregistré, ses états passent à « annulé ».

```
create or replace PROCEDURE PRO_RG_COUPON_ANNULE
(
  I_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
CURSOR CUR_COUPON IS
    SELECT COUPNUM
    FROM COUPON_VOL
    WHERE OCCNUM=I_OCCNUM
    AND COUPETAT<>'enregistré';
BEGIN
    --On pass tous les coupon de vol dont état n'est pas enregistré de cette occurrence de vol dans l'état <annulé>
    FOR V_COUPONS IN CUR_COUPON LOOP

        UPDATE COUPON_VOL SET COUPETAT ='annulé' WHERE COUPNUM=V_COUPONS.COUPNUM;
    END LOOP;
    COMMIT;
END PRO_RG_COUPON_ANNULE;
```

### 3.6. FUNC_NEXTOCCURRENCE

Cette fonction rend la prochaine occurrence de vol selon une occurrence de vol donnée.

```
create or replace FUNCTION FUNC_NEXTOCCURRENCE
(
  P_OCCNUM IN OCCURRENCE_VOL.OCCNUM%TYPE
, P_BILLNUM IN BILLET.BILLNUM%TYPE
) RETURN OCCURRENCE_VOL.OCCNUM%TYPE  AS
  V_NEXTOCCU NUMBER:=-1;
BEGIN
    BEGIN
        --Trouver l'occurrence de vol prochaine dans le trajet
        SELECT DISTINCT(OCC.OCCNUM) INTO V_NEXTOCCU
        FROM CONSTITUER C, OCCURRENCE_VOL OCC, BILLET BIL,
            (SELECT C1.NUMORDRE AS OCCNOW, C1.JOURPLUS AS JP
            FROM CONSTITUER C1, OCCURRENCE_VOL OCC1, BILLET BIL1
            WHERE BIL1.TRANUM = C1.TRANUM
            AND C1.VOLNUM = OCC1.VOLNUM
            AND OCC1.OCCNUM = P_OCCNUM
            AND BIL1.BILLNUM = P_BILLNUM) OCCM
        WHERE BIL.TRANUM = C.TRANUM
        AND C.NUMORDRE = OCCM.OCCNOW + 1
        AND C.VOLNUM = OCC.VOLNUM
        AND TO_DATE(To_char(OCC.OCCDATE, 'YYYY/MM/DD'),'YYYY/MM/DD') = TO_DATE(To_char(BIL.BILLDATEDEPART+C.JOURPLUS,'YYYY/MM/DD'), 'YYYY/MM/DD');

    EXCEPTION
        WHEN NO_DATA_FOUND THEN V_NEXTOCCU:=NULL;
    END;
    RETURN V_NEXTOCCU;
END FUNC_NEXTOCCURRENCE;
```

## 4. Gestion de bagages

### 4.1. PRO_AJOUTER_BAGAGE

Pour enregistrer un bagage, il faut appeler la procédure PRO_AJOUTER_BAGAGE avec un numéro du billet du client et le poids de ce bagage.

La procédure d'abord trouve la première occurrence de vol de ce billet.

Ensuite, elle regarde l'état de cette occurrence de vol. Si l'état est « ouverte à l'embarquement », ce bagage est inséré dans la table « BAGAGE » et cette procédure appelle la procédure PRO_RG_PROVISION_BAGAGE pour la gestion financier et la procédure PRO_RG_AFFECTER pour la gestion d'affectation. Si l'état de l'occurrence de vol n'est pas « ouverte à l'embarquement », ce bagage ne peut pas être enregistré.

```
create or replace PROCEDURE PRO_AJOUTER_BAGAGGE (
    P_BILLNUM   IN BAGAGE.BILLNUM%TYPE,
    P_BAGKG     IN BAGAGE.BAGKG%TYPE
) AS
    V_OCCNUM    OCCURRENCE_VOL.OCCNUM%TYPE;
    V_OCCETAT   OCCURRENCE_VOL.OCCETAT%TYPE;
    V_CLINUM    CLIENT.CLINUM%TYPE;
BEGIN
  --TROUVER LA PREMIÈRE OCCURRENCE POUR CE BILLET
    SELECT OCC.OCCNUM, OCC.OCCETAT INTO V_OCCNUM,V_OCCETAT
      FROM CONSTITUER C, OCCURRENCE_VOL OCC, BILLET BIL
      WHERE BIL.TRANUM = C.TRANUM
      AND C.NUMORDRE = 1
      AND C.VOLNUM = OCC.VOLNUM
      AND TO_DATE(OCC.OCCDATE,'DD/MM/YYYY') = TO_DATE(BIL.BILLDATEDEPART,'DD/MM/YYYY')
      AND BIL.BILLNUM = P_BILLNUM;
    --VÉRIFIER L'ÉTAT DE CE OCCURRENCE
    --Si cette occurrence est en état de "ouverte à l'embarquement"
    IF V_OCCETAT = 'ouverte à l''embarquement' THEN
      --Ajouter ce bagae
      INSERT INTO BAGAGE VALUES (SEQ_BAGAGE.NEXTVAL, P_BILLNUM, P_BAGKG);
      --Créditer ce bagage
      PRO_RG_PROVISION_BAGAGE(P_BILLNUM,P_BAGKG);
      --AFFECTER CE BAGAGE
      PRO_RG_AFFECTER(SEQ_BAGAGE.CURRVAL,NULL,V_OCCNUM);
    ELSE
        RAISE_APPLICATION_ERROR(-20001,'L''occurrence n''est pas ouverte à l''embarquement!');
    END IF;
    --Affichier le résultat d'ajoute
    SELECT CLINUM INTO V_CLINUM
      FROM BILLET
      WHERE BILLNUM = P_BILLNUM;
    DBMS_OUTPUT.PUT_LINE('Le client '||V_CLINUM||' a enregistré un '||P_BAGKG||' kg bagage (N°'||SEQ_BAGAGE.CURRVAL||')');
END PRO_AJOUTER_BAGAGGE;
```

### 4.2. PRO_RG_AFFECTER

Pour affecter un bagage, il faut trois paramètres : numéro du bagage, numéro de l'occurrence provenir, numéro de l'occurrence destiner.

Dans cette procédure, nous trouvons d'abord l'aéroport où ce bagage va être chargé et déchargé. Une fois nous avons trouvé l'aéroport, nous trouvons la limite de poids des conteneurs.

Ensuite, nous regardons s'il y a des conteneurs qui ont encore de place pour mettre ce bagage. Si oui, nous affectons ce bagage au conteneur virtuel trouvé. Si non, nous créons un nouveau conteneur virtuel et affecter ce bagage à ce nouveau virtuel.

```
create or replace PROCEDURE PRO_RG_AFFECTER
(
  I_BAGNUM IN BAGAGE.BAGNUM%TYPE
, I_OCCNUM_PROVENIR IN OCCURRENCE_VOL.OCCNUM%TYPE
, I_OCCNUM_DESTINER IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
  V_MAXCONT AEROPORT.AEROPOIDSMAXCONTAINER%TYPE;
  V_CONTNUM CONTAINER_VIRTUEL.CONTNUM%TYPE;
  V_AERONUM AEROPORT.AERONUM%TYPE;
  V_BAGKG BAGAGE.BAGKG%TYPE;
V_CONTREEL CONTAINER_REEL.CONTNUM%TYPE;
```

```
BEGIN
    /*Trouver les aéroport où des bagages sont.
    (ça peut être l'aéroport d'arrivée de l'occurrence provenier ou/et l'aéroport de départ de l'occurrence destiner)*/
    BEGIN
        SELECT AE.AEROPOIDSMAXCONTAINER, AE.AERONUM INTO V_MAXCONT, V_AERONUM
            FROM AEROPORT AE, OCCURRENCE_VOL OCC, VOL V
            WHERE AE.AERONUM = V.AERONUM_ARRIVEE
            AND V.VOLNUM = OCC.VOLNUM
            AND OCC.OCCNUM = I_OCCNUM_PROVENIR;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            SELECT AE.AEROPOIDSMAXCONTAINER, AE.AERONUM INTO V_MAXCONT, V_AERONUM
                FROM AEROPORT AE, OCCURRENCE_VOL OCC, VOL V
                WHERE AE.AERONUM = V.AERONUM_DEPART
                AND V.VOLNUM = OCC.VOLNUM
                AND OCC.OCCNUM = I_OCCNUM_DESTINER;
    END;
    --Trouver un conteneur réel disponible dans cet aéroport
    V_CONTREEL:=FUNC_AFFECT_REAL(V_AERONUM);
    --Si l'occurrence de vol provenir est null -> C'est le première occurrence de vol
    IF I_OCCNUM_PROVENIR IS NULL THEN
        BEGIN
            --Si il y a un conteneur virtuel qui a assez de place, rendre son numéro
            SELECT BAGKG INTO V_BAGKG FROM BAGAGE WHERE BAGNUM = I_BAGNUM;
            SELECT C.CONTNUM INTO V_CONTNUM
                FROM CONTAINER_VIRTUEL C, BAGAGE B, AFFECTER A
                WHERE C.CONTNUM = A.CONTNUM
                AND A.BAGNUM = B.BAGNUM
                AND C.OCCNUM_PROVENIR IS NULL
                AND C.OCCNUM_DESTINER = I_OCCNUM_DESTINER
                AND ROWNUM = 1
                GROUP BY C.CONTNUM
                HAVING SUM(B.BAGKG)+V_BAGKG< V_MAXCONT;
        EXCEPTION
            --Si non, il faut créer un conteneur virtuel
            WHEN NO_DATA_FOUND THEN
                INSERT INTO CONTAINER_VIRTUEL VALUES(SEQ_CONTAINER.NEXTVAL, I_OCCNUM_DESTINER, NULL, NULL, V_CONTREEL);
                V_CONTNUM := SEQ_CONTAINER.CURRVAL;
        END;
    --Si l'occurrence de vol destiner est null -> C'est le dernière occurrence de vol
    ELSIF I_OCCNUM_DESTINER IS NULL THEN
        BEGIN
            --Si il y a un conteneur virtuel qui a assez de place, rendre son numéro
            SELECT BAGKG INTO V_BAGKG FROM BAGAGE WHERE BAGNUM = I_BAGNUM;
            SELECT C.CONTNUM INTO V_CONTNUM
                FROM CONTAINER_VIRTUEL C, BAGAGE B, AFFECTER A
                WHERE C.CONTNUM = A.CONTNUM
                AND A.BAGNUM = B.BAGNUM
                AND C.OCCNUM_PROVENIR = I_OCCNUM_PROVENIR
                AND C.OCCNUM_DESTINER IS NULL
                AND ROWNUM = 1
                GROUP BY C.CONTNUM
                HAVING SUM(B.BAGKG)+V_BAGKG< V_MAXCONT;
        EXCEPTION
            --Si non, il faut créer un conteneur virtuel
            WHEN NO_DATA_FOUND THEN
                INSERT INTO CONTAINER_VIRTUEL VALUES(SEQ_CONTAINER.NEXTVAL, NULL, I_OCCNUM_PROVENIR, NULL, V_CONTREEL);
                V_CONTNUM := SEQ_CONTAINER.CURRVAL;
        END;
```

```
    --Si l'occurrence de vol destiner est null -> C'est le dernière occurrence de vol
    ELSE
        BEGIN
            --Si il y a un conteneur virtuel qui a assez de place, rendre son numéro
            SELECT BAGKG INTO V_BAGKG FROM BAGAGE WHERE BAGNUM = I_BAGNUM;
            SELECT C.CONTNUM INTO V_CONTNUM
                FROM CONTAINER_VIRTUEL C, BAGAGE B, AFFECTER A
                WHERE C.CONTNUM = A.CONTNUM
                AND A.BAGNUM = B.BAGNUM
                AND C.OCCNUM_PROVENIR = I_OCCNUM_PROVENIR
                AND C.OCCNUM_DESTINER = I_OCCNUM_DESTINER
                AND ROWNUM = 1
                GROUP BY C.CONTNUM
                HAVING SUM(B.BAGKG)+V_BAGKG< V_MAXCONT;
        EXCEPTION
            --Si non, il faut créer un conteneur virtuel
            WHEN NO_DATA_FOUND THEN
            SYS.DBMS_OUTPUT.PUT_LINE('destiner---'||I_OCCNUM_DESTINER||'--provenir----'||I_OCCNUM_PROVENIR||'---cont reel'||V_CONTREEL);
            INSERT INTO CONTAINER_VIRTUEL VALUES(SEQ_CONTAINER.NEXTVAL, I_OCCNUM_DESTINER, I_OCCNUM_PROVENIR, NULL, V_CONTREEL);
            V_CONTNUM := SEQ_CONTAINER.CURRVAL;
        END;
    END IF;
    --Affecter lr bagage au conteneur trouvé
    INSERT INTO AFFECTER VALUES(I_BAGNUM, V_CONTNUM, SYSDATE);
    COMMIT;
END PRO_RG_AFFECTER;
```

### *4.3. PRO_CHARGER*

Cette procédure est pour le chargement. Pour charger un bagage, des employés doivent scanner les codes-barres (numéro de bagage) et saisir le numéro de conteneur virtuel.

Avec les informations, nous vérifie d'abord si le couple BAGNUM – CONTNUM existe dans la table « affecter », si non, ce bagage n'est pas affecté à ce conteneur, la procédure s'arrête.

```
create or replace PROCEDURE PRO_CHARGER
(
    P_BAGNUM    IN BAGAGE.BAGNUM%TYPE
,   P_CONTNUM   IN CONTAINER_VIRTUEL.CONTNUM%TYPE
) AS
    V_AFFECTE   NUMBER;
    V_NBAFFEC   NUMBER;
    V_NBCHAR    NUMBER;
    V_OCC_PROV_ETAT OCCURRENCE_VOL.OCCETAT%TYPE;
    V_OK BOOLEAN := FALSE;
    V_FLAG_PREMIER_OCC OCCURRENCE_VOL.OCCNUM%TYPE;
BEGIN
    --Vérifier si ce bagage est correspondant à le container
    SELECT COUNT(*) INTO V_AFFECTE FROM AFFECTER WHERE BAGNUM = P_BAGNUM AND CONTNUM = P_CONTNUM;
    IF V_AFFECTE = 0 THEN
        --SI CE BAGAGE N'EST PAS AFFECTÉ À CE CONTENEUR -> ARRÊTER LE PROCÉDURE
        RAISE_APPLICATION_ERROR(-20001, 'Ce bagage n''est pas affecté à ce conteneur!');
    ELSE
    --Vérifier si ce container correspond une occurrence provenu
    SELECT CASE
        WHEN OCCNUM_PROVENIR IS NULL THEN -1 ELSE OCCNUM_PROVENIR END AS OCCNUM_PROVENIR
        INTO V_FLAG_PREMIER_OCC
        FROM CONTAINER_VIRTUEL
        WHERE CONTNUM=P_CONTNUM;
    --Si V_FLAG_PREMIER_OCC n'est pas -1, c'est à dire ce container correspond une occurrence provenu
    IF V_FLAG_PREMIER_OCC<>-1 THEN
    --Trouver l'état de l'occurrence de vol provenu pour véfirifier si le container peut être chargé
        SELECT
            CASE WHEN OCC.OCCETAT IS NULL THEN 'premier' ELSE OCC.OCCETAT END AS OCCETAT
            INTO V_OCC_PROV_ETAT
            FROM OCCURRENCE_VOL OCC, CONTAINER_VIRTUEL C
            WHERE C.OCCNUM_PROVENIR = OCC.OCCNUM
            AND C.CONTNUM = P_CONTNUM;
            /*Le container peut être chargé quant l'état de l'occurrence de vol provenu est seulment
            arrivée ou déroutée ou c'est la prémière occurrence de vol dans le trajet*/
            IF V_OCC_PROV_ETAT <> 'arrivée' AND V_OCC_PROV_ETAT<>'déroutée' AND V_OCC_PROV_ETAT<>'premier' THEN
                RAISE_APPLICATION_ERROR(-20003,
                    'vous ne pouvez pas charger des bagages d''une occurrence qui n''est pas encore arrivée!');
            END IF;
    END IF;
        --Charger les bagages
        INSERT INTO CHARGER VALUES(P_BAGNUM, P_CONTNUM, SYSDATE);
        --Si on charge le premier bagage, on pass le container virtuel dans l'état <en cours de chargement>.
        SELECT COUNT(*) INTO V_NBAFFEC FROM AFFECTER WHERE CONTNUM = P_CONTNUM;
        SELECT COUNT(*) INTO V_NBCHAR FROM CHARGER WHERE CONTNUM = P_CONTNUM;
        IF V_NBCHAR = 1 THEN
            UPDATE CONTAINER_VIRTUEL SET CONTETAT = 'en cours de chargement' WHERE CONTNUM = P_CONTNUM;
        END IF;
        --Si tous les bagaes sont bien chargés, on pass le container virtuel dans l'état <chargé>.
        IF V_NBCHAR = V_NBAFFEC THEN
            UPDATE CONTAINER_VIRTUEL SET CONTETAT = 'chargé' WHERE CONTNUM = P_CONTNUM;
        END IF;
    END IF;
END PRO_CHARGER;
```

### 4.4. PRO_DECHARGER

Dans cette procédure, nous regardons l'état de conteneur virtuel, Un bagage peut seulement être déchargé si l'état de container est <chargé> ou <en cours de déchargement>. Si c'est le cas, nous ajoutons les informations de déchargement dans la table « decharger ».

Ensuite, nous compter le nombre de bagages affectés à ce conteneur et le nombre de bagages déchargés de ce conteneur. Pour savoir si ce bagage est le premier ou le dernier bagage déchargé de ce conteneur. Si c'est le premier bagage, nous passons ce container dans l'état < en cours de déchargement > ; Si tous les bagages sont bien déchargés, nous passons ce container dans l'état <déchargé>.

```
create or replace PROCEDURE PRO_DECHARGER (
    P_BAGNUM    IN BAGAGE.BAGNUM%TYPE,
    P_CONTNUM   IN CONTAINER_VIRTUEL.CONTNUM%TYPE
) AS
    V_AFFECTE    NUMBER;
    V_NBAFFEC    NUMBER;
    V_NBDECHAR   NUMBER;
    V_ETAT       CONTAINER_VIRTUEL.CONTETAT%TYPE;
    V_CONTREEL   CONTAINER_REEL.CONTNUM%TYPE;
BEGIN
    --Trouber l'état de container virtuel
    SELECT CONTETAT INTO V_ETAT FROM CONTAINER_VIRTUEL WHERE CONTNUM = P_CONTNUM;
    --Un bagage peut seulment être déchargé si l'état de container est <chargé> ou <en cours de déchargement>
    IF V_ETAT = 'chargé' OR V_ETAT = 'en cours de déchargement' THEN
        --Vérifier si ce bagage peut être affecté à ce conteneur
        SELECT COUNT(*) INTO V_AFFECTE FROM AFFECTER WHERE BAGNUM = P_BAGNUM AND   CONTNUM = P_CONTNUM;
        IF V_AFFECTE = 0 THEN
            RAISE_APPLICATION_ERROR(-20001,'Ce bagage n''est pas affecté à ce conteneur!');
        ELSE
            --Décharger ce bagage
            INSERT INTO DECHARGER VALUES (P_BAGNUM,P_CONTNUM,SYSDATE);
            --Si on décharge à la première fois un bagage, on passe ce container dans l'état <en cours de déchargement>
            SELECT COUNT(*) INTO V_NBAFFEC FROM AFFECTER WHERE CONTNUM = P_CONTNUM;
            SELECT COUNT(*) INTO V_NBDECHAR FROM DECHARGER WHERE CONTNUM = P_CONTNUM;
            IF V_NBDECHAR = 1 THEN
                UPDATE CONTAINER_VIRTUEL SET CONTETAT = 'en cours de déchargement'
                    WHERE CONTNUM = P_CONTNUM;
            END IF;
            --Si tous les bagages sont bien déchargés, on pass ce container dans l'état <déchargé>
            IF V_NBDECHAR = V_NBAFFEC THEN
                SELECT CONTREEL INTO V_CONTREEL FROM CONTAINER_VIRTUEL WHERE CONTNUM = P_CONTNUM;
                UPDATE CONTAINER_VIRTUEL SET CONTETAT = 'déchargeé' WHERE CONTNUM = P_CONTNUM;
                UPDATE CONTAINER_REEL SET DISPONIBLE ='oui' WHERE CONTNUM=V_CONTREEL;
            END IF;
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20001,'Ce conteneur n''est pas encore chargé!');
    END IF;
EXCEPTION
    --Si on ne trouve pas l'état de container, c'est-à-dire Ce conteneur virtuel n'existe pas!
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001,'Ce conteneur virtuel n''existe pas!');
END PRO_DECHARGER;
```

### *4.5. PRO_DEROUTER_AFFECTER*

Si une occurrence de vol est déroutée, il faut réaffecter tous les bagages avec cette procédure.

Nous trouvons tous les conteneurs virtuels correspondants à l'occurrence déroutée. Après, nous supprimons tous les bagages affectés à ces conteneurs et utilisons la procédure PRO_RG_AFFECTER pour affecter ces bagages à des nouveaux conteneurs.

```
create or replace PROCEDURE PRO_DEROUTER_AFFECTER
(
  P_BAGNUM          IN BAGAGE.BAGNUM%TYPE
, P_OCCNUM_DESTINER IN OCCURRENCE_VOL.OCCNUM%TYPE
, P_OCCNUM_DEROUTER IN OCCURRENCE_VOL.OCCNUM%TYPE
, P_OCCNUM_PROVENIR IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
--Trouver les  containers anciens
CURSOR CUR_CONT IS
    SELECT CONTNUM
    FROM  CONTAINER_VIRTUEL
    WHERE OCCNUM_PROVENIR=P_OCCNUM_PROVENIR
    AND OCCNUM_DESTINER=P_OCCNUM_DESTINER;
BEGIN
    --SUPPRIMER LES AFFECTION ANCIENES
    FOR V_CONT IN CUR_CONT LOOP
        --SUPPRIMER LES LIGNES DANS LA TABLE AFFECTER
        DELETE FROM AFFECTER WHERE CONTNUM= V_CONT.CONTNUM;
        --SUPPRIMER LES CONTAINERS
    DELETE FROM CONTAINER_VIRTUEL WHERE CONTNUM= V_CONT.CONTNUM;
    END LOOP;
    --AJOUTER NOUVEL AFFECTER
    PRO_RG_AFFECTER(P_BAGNUM,P_OCCNUM_PROVENIR,P_OCCNUM_DEROUTER);
END PRO_DEROUTER_AFFECTER;
```

```
create or replace PROCEDURE PRO_DEROUTER_AFFECTER
(
  P_BAGNUM          IN BAGAGE.BAGNUM%TYPE
, P_OCCNUM_DESTINER IN OCCURRENCE_VOL.OCCNUM%TYPE
, P_OCCNUM_DEROUTER IN OCCURRENCE_VOL.OCCNUM%TYPE
, P_OCCNUM_PROVENIR IN OCCURRENCE_VOL.OCCNUM%TYPE
) AS
--Trouver les  containers anciens
CURSOR CUR_CONT IS
    SELECT CONTNUM
    FROM  CONTAINER_VIRTUEL
    WHERE OCCNUM_PROVENIR=P_OCCNUM_PROVENIR
    AND OCCNUM_DESTINER=P_OCCNUM_DESTINER;
BEGIN
    --SUPPRIMER LES AFFECTION ANCIENES
    FOR V_CONT IN CUR_CONT LOOP
        --SUPPRIMER LES LIGNES DANS LA TABLE AFFECTER
        DELETE FROM AFFECTER WHERE CONTNUM= V_CONT.CONTNUM;
        --SUPPRIMER LES CONTAINERS
    DELETE FROM CONTAINER_VIRTUEL WHERE CONTNUM= V_CONT.CONTNUM;
    END LOOP;
    --AJOUTER NOUVEL AFFECTER
    PRO_RG_AFFECTER(P_BAGNUM,P_OCCNUM_PROVENIR,P_OCCNUM_DEROUTER);
END PRO_DEROUTER_AFFECTER;
```

## 5. Gestion financière

### 5.1. PRO_RG_PROVISION_BAGAGE

A l'enregistrement des bagages, on doit créditer sur le compte de provision le montant pour la surcharge des bagages de ce client.

Nous trouvons d'abord la limite de poids d'un billet. Ensuite, nous voyons le tableau provision pour vérifier si le client a déjà payé pour son bagage. S'il n'y a pas de provision crédit pour ces bagages, nous comptons la somme de poids des bagages pour vérifier s'il faut que le client paie pour la surcharge. Si le client a déjà payé pour la surcharge de bagages, le client doit payer un montant calculé avec tout le poids du nouveau bagage.

```
create or replace PROCEDURE PRO_RG_PROVISION_BAGAGE
(
  P_BILLNUM IN BILLET.BILLNUM%TYPE
, P_BAGKG IN BAGAGE.BAGKG%TYPE
) AS
  V_LIMITEKG TRAJET.TRANKGBAG%TYPE;
  V_KGBAG TRAJET.TRANKGBAG%TYPE;
  V_TRATARIFKGSUP TRAJET.TRATARIFKGSUP%TYPE;
  V_MONTANT_BAG PROVISION.PROVMONTANT%TYPE;
  V_NBOPPROV NUMBER;
BEGIN
    --Trouver le poids limité pout ce billet
    SELECT T.TRANKGBAG, T.TRATARIFKGSUP INTO V_LIMITEKG, V_TRATARIFKGSUP
        FROM TRAJET T, BILLET BIL
        WHERE BIL.TRANUM = T.TRANUM
        AND BIL.BILLNUM = P_BILLNUM;
    --Voir le tableau povision pour vérifier si le client a déjà payé pour son bagage
    SELECT COUNT (*) INTO V_NBOPPROV
        FROM PROVISION
        WHERE BILLNUM = P_BILLNUM
        AND PROVOPTYPE='crédit';
    --If no extra fee, count all bagages to see if need to pay more
    IF V_NBOPPROV = 1 THEN
        --Count the weight of all bagages of this ticket
        SELECT NVL(SUM(BAGKG),0) INTO V_KGBAG
            FROM BAGAGE
            WHERE BILLNUM = P_BILLNUM;
    --Comparer le poids du bagage et le poids limité pour ce billet
        IF V_LIMITEKG < (V_KGBAG + P_BAGKG) THEN
            --Compter le montant suplimentaire et ajouter ce montant dans le tableau provision
            V_MONTANT_BAG := (V_KGBAG + P_BAGKG - V_LIMITEKG) * V_TRATARIFKGSUP;
            INSERT INTO PROVISION VALUES(SEQ_PROVISION.NEXTVAL, P_BILLNUM, 'crédit', V_MONTANT_BAG, SYSDATE);
        END IF;
    ELSE
        V_MONTANT_BAG := P_BAGKG * V_TRATARIFKGSUP;
        INSERT INTO PROVISION VALUES(SEQ_PROVISION.NEXTVAL, P_BILLNUM, 'crédit', V_MONTANT_BAG, SYSDATE);
    END IF;
  COMMIT;
END PRO_RG_PROVISION_BAGAGE;
```

### 5.2. PRO_RG_PROVISION_TICKET

Pour faire une provision de ticket, nous trouvons le tarif du trajet correspondant et le taxe de l'aéroport. La provision de ticket est la somme des deux parties.

```
create or replace PROCEDURE PRO_RG_PROVISION_TICKET
(
  P_BILLETNUM IN BILLET.BILLNUM%TYPE,
  P_TRAJETNUM IN TRAJET.TRANUM%TYPE
) AS
    V_MONTANT_TRAJET NUMBER;
    V_MONTANT_TAXE AEROPORT.AEROTAXE%TYPE;
BEGIN
    --Calcul le montant de trajet
    SELECT TRATARIFBILLET INTO V_MONTANT_TRAJET FROM TRAJET WHERE TRANUM=P_TRAJETNUM;
    --Calcul le somme de taxe
    SELECT SUM(AEROTAXE) INTO V_MONTANT_TAXE
        FROM(--La liste de taxe
            SELECT AEROTAXE
                FROM CONSTITUER C,VOL V,AEROPORT A
                WHERE C.TRANUM=P_TRAJETNUM
                AND C.VOLNUM=V.VOLNUM
                AND (V.AERONUM_DEPART=A.AERONUM
                OR   V.AERONUM_ARRIVEE=A.AERONUM)
        ) A;
    --Inserter la valeur
    INSERT INTO PROVISION VALUES(SEQ_PROVISION.NEXTVAL,P_BILLETNUM,'crédit',V_MONTANT_TRAJET+V_MONTANT_TAXE,SYSDATE);
END PRO_RG_PROVISION_TICKET;
```

### 5.3. TRIG_BILLET_FINANCE

Lorsque l'état d'un billet passe à « terminé », nous comptons la somme de toutes les provisions de type de crédit de ce billet. Nous débitons la somme dans la table « provision » et la créditons dans la table « recette ».

```
create or replace TRIGGER TRIG_BILLET_FINANCE
BEFORE UPDATE OF BILLETAT ON BILLET
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
WHEN (N.billetat='terminé')
DECLARE
v_montant number;
BEGIN
  select sum(provmontant) INTO v_montant
  FROM PROVISION
  WHERE BILLNUM=:N.BILLNUM
  AND PROVOPTYPE='crédit';

  insert into PROVISION values(SEQ_PROVISION.nextVal,:N.BILLNUM,'dédit',v_montant,sysdate);
  insert into RECETTE values(SEQ_RECETTE.nextVal,:N.BILLNUM,v_montant,sysdate);

END;
```

## 6. Gestion de conteneurs

### 6.1. FUNC_AFFECT_REAL

Cette fonction trouve un conteneur réel disponible dans l'aéroport. S'il n'y a pas de résultat de la requête, un message « il n'y a pas assez de conteneur réel » s'affiche.

```
create or replace FUNCTION FUNC_AFFECT_REAL (p_aeronum IN VARCHAR2)
RETURN NUMBER
AS
  v_result NUMBER;
BEGIN
    SELECT CONTNUM
    INTO   v_result
    FROM   CONTAINER_REEL
    WHERE  AERONUM = P_AERONUM
           AND ROWNUM = 1
           AND CONTNUM NOT IN(SELECT CONTREEL
                              FROM   CONTAINER_VIRTUEL);
    dbms_output.PUT_LINE(V_RESULT);
    RETURN V_RESULT;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20200,
            'Il n''y pas assez de container réel');
END func_affect_real;
```

### 6.2. TRI_CONT_REEL

Lorsqu'un conteneur virtuel est créé, la disponibilité du conteneur réel qui est affecté à ce conteneur virtuel passe à « non ».

```
create or replace TRIGGER "TRI_CONT_REEL" BEFORE INSERT ON CONTAINER_VIRTUEL
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
BEGIN
  UPDATE CONTAINER_REEL SET DISPONIBLE='non' WHERE CONTNUM=:N.CONTREEL;
END;
```

### 6.3. TRIG_LIBRE_CON

Lorsque l'état d'un conteneur virtuel passe à « déchargé », la disponibilité du conteneur réel correspondant passe à « oui ».

```
create or replace TRIGGER TRIG_LIBRE_CON
BEFORE UPDATE OF CONTETAT ON CONTAINER_VIRTUEL
REFERENCING OLD AS O NEW AS N
FOR EACH ROW
WHEN (N.CONTETAT='déchargé')
BEGIN
    UPDATE CONTAINER_REEL SET DISPONIBLE='oui' WHERE CONTNUM=:N.CONTREEL;
END;
```