



# GESTION DE COMMANDE DE CENTRALE D'ACHAT

PL/SQL ORACLE



**Xiyue LIAO**

**Yanrui GUO**

13/01/2019

# SOMMAIRE

<b>PARTIE I : CREATION ET ADMINISTRATION DE LA BASE DE DONNEES .....</b>	<b>2</b>
1-1 : RETRO-CONCEPTION DU SCHEMA ENTITE/ASSOCIATION.....	2
1-2 : CREATION LA BASE DE DONNEES ET DEUX UTILISATEURS.....	4
1-2-1 : <i>Création de la base de données</i> .....	4
1-2-2 : <i>Création des utilisateurs</i> .....	14
<b>PARTIE II : REDACTION D'UN DOSSIER DE SPECIFICATIONS FONCTIONNELLES.....</b>	<b>18</b>
2-1 : HYPOTHESES ET REGLES DE GESTION CONCERNANT LA DYNAMIQUE .....	18
2-1-1 : <i>Hypothèses</i> .....	18
2-1-2 : <i>Règles de gestion concernant la dynamique</i> .....	18
2-1-2-1 : Création des chargements.....	19
2-1-2-2 : Création des commandes .....	19
2-1-2-3 : Factorisation des commandes.....	20
2-1-2-4 : Préparation de commande globale et création des bons livraison.....	21
2-1-2-5 : Contrôle de la quantité de produit livré .....	21
2-1-2-6 : Distribution des colis .....	22
2-2 : DECOMPOSITION MODULAIRE.....	24
2-2-1 : <i>Procédures</i> .....	24
2-2-1-1 : RG_PREPARATION_COMMANDE .....	24
2-2-1-2 : RG_FACTORISATION_COMMANDE .....	24
2-2-1-3 : RG_TERMINER_COMMANDE .....	27
2-2-1-4 : RG_PROD_QTE_LIV .....	27
2-2-1-5 : CREATION_CHARGEMENT .....	29
2-2-1-6 : RG_DISTRIBUTION_COLIS.....	30
2-2-1-7 : COLIS_LIVRE .....	31
2-2-2 : <i>Déclencheurs</i> .....	32
2-2-2-1 : CREATION_COMMANDE.....	32
2-2-2-2 : INSERER_LIGNE_COMMANDE.....	32
2-2-2-3 : INSERER_LIGNE_COMGLOB.....	32
2-2-2-4 : FACTORISATION_COMMANDE.....	33
2-2-2-5 : BONLIVNUM_AUTO.....	33
2-2-2-6 : CONTROLE_QTE_BONLIV.....	33
2-2-2-7 : CHARGEMENTNUM_AUTO.....	33
2-2-2-8 : LIVRAISON_COLIS .....	34
2-2-2-9 : EXPEDIER_CHARGEMENT .....	34
2-2-3 : <i>Fonctions</i> .....	35
2-2-3-1 : RG_QTE_COMMANDE .....	35
2-2-4 : <i>Séquences</i> .....	35
2-2-4-1 : SEQ_CHARGEMENT .....	35
2-2-4-2 : SEQ_COMMANDE.....	35
2-2-4-3 : SEQ_COMGLOB.....	35
2-2-4-4 : SEQ_COLIS .....	35
2-2-4-5 : SEQ_BONLIV .....	35
<b>PARTIE III : PROGRAMMATION PL/SQL .....</b>	<b>36</b>
3-1 : ENVIRONNEMENT DE TEST .....	36
3-2 : RESULTAT DE TEST.....	39
3-2-1 <i>Préparation de test</i> .....	39
3-2-2 <i>Factorisation</i> .....	40
3-2-3 <i>Bonlivraison</i> .....	44
3-2-4 <i>Chargement et Livraison</i> .....	47
<b>CONCLUSION .....</b>	<b>51</b>





## 1-2 : Création la base de données et deux utilisateurs

### 1-2-1 : Création de la base de données

Depuis le schéma que nous avons créé sur Win'Design, nous obtenons les codes pour la constitution de la base de données.

Voici les codes de création de la base de données.

```
1. DROP TABLE SECTEUR CASCADE CONSTRAINTS;
2.
3. DROP TABLE FOURNISSEUR CASCADE CONSTRAINTS;
4.
5. DROP TABLE BONLIVRAISON CASCADE CONSTRAINTS;
6.
7. DROP TABLE COMMANDE CASCADE CONSTRAINTS;
8.
9. DROP TABLE COMMANDEGLOBALE CASCADE CONSTRAINTS;
10.
11. DROP TABLE CAMION CASCADE CONSTRAINTS;
12.
13. DROP TABLE PRODUIT CASCADE CONSTRAINTS;
14.
15. DROP TABLE MODELE CASCADE CONSTRAINTS;
16.
17. DROP TABLE MAGASIN CASCADE CONSTRAINTS;
18.
19. DROP TABLE CHARGEMENT CASCADE CONSTRAINTS;
20.
21. DROP TABLE COLIS CASCADE CONSTRAINTS;
22.
23. DROP TABLE LIVRER CASCADE CONSTRAINTS;
24.
25. DROP TABLE CONCERNER CASCADE CONSTRAINTS;
26.
27. DROP TABLE ASSOCIER CASCADE CONSTRAINTS;
28.
29. DROP TABLE STOCKER CASCADE CONSTRAINTS;
30.
31. DROP TABLE FOURNIR CASCADE CONSTRAINTS;
32.
33. DROP TABLE CONCERNERGLOB CASCADE CONSTRAINTS;
34.
35. -----
36. --      CREATION DE LA BASE
37. -----
38.
39. CREATE DATABASE MLR2;
40.
41. -----
42. --      TABLE : SECTEUR
43. -----
```

```
44.
45. CREATE TABLE SECTEUR
46. (
47.   SECNUM CHAR(32) NOT NULL,
48.   SECVILLE CHAR(32) NULL,
49.   SECQUARTIER CHAR(32) NULL
50. , CONSTRAINT PK_SECTEUR PRIMARY KEY (SECNUM)
51. );
52.
53. -----
54. --   TABLE : FOURNISSEUR
55. -----
56.
57. CREATE TABLE FOURNISSEUR
58. (
59.   FOURNUM CHAR(32) NOT NULL,
60.   FOURNOM CHAR(32) NULL
61. , CONSTRAINT PK_FOURNISSEUR PRIMARY KEY (FOURNUM)
62. );
63.
64. -----
65. --   TABLE : BONLIVRAISON
66. -----
67.
68. CREATE TABLE BONLIVRAISON
69. (
70.   BONLIVNUM CHAR(32) NOT NULL,
71.   COMGLOBNUM CHAR(32) NOT NULL,
72.   BONDATE CHAR(32) NULL
73. , CONSTRAINT PK_BONLIVRAISON PRIMARY KEY (BONLIVNUM)
74. );
75.
76. -----
77. --   INDEX DE LA TABLE BONLIVRAISON
78. -----
79.
80. CREATE INDEX I_FK_BONLIVRAISON_COMMANDEGLOB
81.   ON BONLIVRAISON (COMGLOBNUM ASC)
82. ;
83.
84. -----
85. --   TABLE : COMMANDE
86. -----
87.
88. CREATE TABLE COMMANDE
89. (
90.   COMNUM CHAR(32) NOT NULL,
91.   MAGNUM CHAR(32) NOT NULL,
92.   COMDATE CHAR(32) NULL,
93.   COMDATELIV CHAR(32) NULL,
94.   COMETAT CHAR(32) NULL
95. , CONSTRAINT PK_COMMANDE PRIMARY KEY (COMNUM)
```

```
96. );
97.
98. -----
99. -- INDEX DE LA TABLE COMMANDE
100. -----
101.
102. CREATE INDEX I_FK_COMMANDE_MAGASIN
103. ON COMMANDE (MAGNUM ASC)
104. ;
105.
106. -----
107. -- TABLE : COMMANDEGLOBALE
108. -----
109.
110. CREATE TABLE COMMANDEGLOBALE
111. (
112. COMGLOBNUM CHAR(32) NOT NULL,
113. FOURNUM CHAR(32) NOT NULL,
114. COMGLOBDATE CHAR(32) NULL,
115. COMGLOBETAT CHAR(32) NULL
116. , CONSTRAINT PK_COMMANDEGLOBALE PRIMARY KEY (COMGLOBNUM)
117. );
118.
119. -----
120. -- INDEX DE LA TABLE COMMANDEGLOBALE
121. -----
122.
123. CREATE INDEX I_FK_COMMANDEGLOBALE_FOURNISSE
124. ON COMMANDEGLOBALE (FOURNUM ASC)
125. ;
126.
127. -----
128. -- TABLE : CAMION
129. -----
130.
131. CREATE TABLE CAMION
132. (
133. CAMNUM CHAR(32) NOT NULL,
134. MODNUM CHAR(32) NOT NULL,
135. CAMKM CHAR(32) NULL
136. , CONSTRAINT PK_CAMION PRIMARY KEY (CAMNUM)
137. );
138.
139. -----
140. -- INDEX DE LA TABLE CAMION
141. -----
142.
143. CREATE INDEX I_FK_CAMION_MODELE
144. ON CAMION (MODNUM ASC)
145. ;
146.
147. -----
```

```
148.  --  TABLE : PRODUIT
149.  -----
150.
151.  CREATE TABLE PRODUIT
152.  (
153.    PRODNUM CHAR(32) NOT NULL,
154.    PRODLIB CHAR(32) NULL,
155.    PRODVOLUME CHAR(32) NULL
156.  , CONSTRAINT PK_PRODUIT PRIMARY KEY (PRODNUM)
157.  );
158.
159.  -----
160.  --  TABLE : MODELE
161.  -----
162.
163.  CREATE TABLE MODELE
164.  (
165.    MODNUM CHAR(32) NOT NULL,
166.    MODLIBELLE CHAR(32) NULL,
167.    MODCAPACITE CHAR(32) NULL
168.  , CONSTRAINT PK_MODELE PRIMARY KEY (MODNUM)
169.  );
170.
171.  -----
172.  --  TABLE : MAGASIN
173.  -----
174.
175.  CREATE TABLE MAGASIN
176.  (
177.    MAGNUM CHAR(32) NOT NULL,
178.    SECNUM CHAR(32) NOT NULL,
179.    MAGNOM CHAR(32) NULL
180.  , CONSTRAINT PK_MAGASIN PRIMARY KEY (MAGNUM)
181.  );
182.
183.  -----
184.  --  INDEX DE LA TABLE MAGASIN
185.  -----
186.
187.  CREATE INDEX I_FK_MAGASIN_SECTEUR
188.    ON MAGASIN (SECNUM ASC)
189.  ;
190.
191.  -----
192.  --  TABLE : CHARGEMENT
193.  -----
194.
195.  CREATE TABLE CHARGEMENT
196.  (
197.    CHARNUM CHAR(32) NOT NULL,
198.    CAMNUM CHAR(32) NULL,
199.    CHARDATE CHAR(32) NULL,
```



```
200.      CHARETAT CHAR(32) NULL
201.      , CONSTRAINT PK_CHARGEMENT PRIMARY KEY (CHARNUM)
202.      );
203.
204.      -----
205.      --      INDEX DE LA TABLE CHARGEMENT
206.      -----
207.
208.      CREATE INDEX I_FK_CHARGEMENT_CAMION
209.      ON CHARGEMENT (CAMNUM ASC)
210.      ;
211.
212.      -----
213.      --      TABLE : COLIS
214.      -----
215.
216.      CREATE TABLE COLIS
217.      (
218.      COLNUM CHAR(32) NOT NULL,
219.      CHARNUM CHAR(32) NULL,
220.      PRODNUM CHAR(32) NULL,
221.      COMNUM CHAR(32) NOT NULL,
222.      COLETAT CHAR(32) NULL,
223.      COLVOLUME CHAR(32) NULL,
224.      QTELIV CHAR(32) NULL
225.      , CONSTRAINT PK_COLIS PRIMARY KEY (COLNUM)
226.      );
227.
228.      -----
229.      --      INDEX DE LA TABLE COLIS
230.      -----
231.
232.      CREATE INDEX I_FK_COLIS_CHARGEMENT
233.      ON COLIS (CHARNUM ASC)
234.      ;
235.
236.      CREATE INDEX I_FK_COLIS_PRODUIT
237.      ON COLIS (PRODNUM ASC)
238.      ;
239.
240.      CREATE INDEX I_FK_COLIS_COMMANDE
241.      ON COLIS (COMNUM ASC)
242.      ;
243.
244.      -----
245.      --      TABLE : LIVRER
246.      -----
247.
248.      CREATE TABLE LIVRER
249.      (
250.      PRODNUM CHAR(32) NOT NULL,
251.      BONLIVNUM CHAR(32) NOT NULL,
```

```
252.      QTELIV CHAR(32) NULL,
253.      QTEREFUS CHAR(32) NULL
254. ,   CONSTRAINT PK_LIVRER PRIMARY KEY (PRODNUM, BONLIVNUM)
255. );
256.
257. -----
258. --   INDEX DE LA TABLE LIVRER
259. -----
260.
261. CREATE INDEX I_FK_LIVRER_PRODUIT
262.     ON LIVRER (PRODNUM ASC)
263. ;
264.
265. CREATE INDEX I_FK_LIVRER_BONLIVRAISON
266.     ON LIVRER (BONLIVNUM ASC)
267. ;
268.
269. -----
270. --   TABLE : CONCERNER
271. -----
272.
273. CREATE TABLE CONCERNER
274. (
275.     COMNUM CHAR(32) NOT NULL,
276.     PRODNUM CHAR(32) NOT NULL,
277.     QTECOM CHAR(32) NULL
278. ,   CONSTRAINT PK_CONCERNER PRIMARY KEY (COMNUM, PRODNUM)
279. );
280.
281. -----
282. --   INDEX DE LA TABLE CONCERNER
283. -----
284.
285. CREATE INDEX I_FK_CONCERNER_COMMANDE
286.     ON CONCERNER (COMNUM ASC)
287. ;
288.
289. CREATE INDEX I_FK_CONCERNER_PRODUIT
290.     ON CONCERNER (PRODNUM ASC)
291. ;
292.
293. -----
294. --   TABLE : ASSOCIER
295. -----
296.
297. CREATE TABLE ASSOCIER
298. (
299.     COMNUM CHAR(32) NOT NULL,
300.     COMGLOBNUM CHAR(32) NOT NULL,
301.     PRODNUM CHAR(32) NOT NULL
302. ,   CONSTRAINT PK_ASSOCIER PRIMARY KEY (COMNUM, COMGLOBNUM, PRODNUM)
303. );
```

```
304.
305.  -----
306.  --  INDEX DE LA TABLE ASSOCIER
307.  -----
308.
309.  CREATE INDEX I_FK_ASSOCIER_COMMANDE
310.    ON ASSOCIER (COMNUM ASC)
311.    ;
312.
313.  CREATE INDEX I_FK_ASSOCIER_COMMANDEGLOBALE
314.    ON ASSOCIER (COMGLOBNUM ASC)
315.    ;
316.
317.  CREATE INDEX I_FK_ASSOCIER_PRODUIT
318.    ON ASSOCIER (PRODNUM ASC)
319.    ;
320.
321.  -----
322.  --  TABLE : STOCKER
323.  -----
324.
325.  CREATE TABLE STOCKER
326.  (
327.    MAGNUM CHAR(32) NOT NULL,
328.    PRODNUM CHAR(32) NOT NULL,
329.    QTESTOCK CHAR(32) NULL,
330.    QTEALERTE CHAR(32) NULL,
331.    QTEMAX CHAR(32) NULL
332.  , CONSTRAINT PK_STOCKER PRIMARY KEY (MAGNUM, PRODNUM)
333.  );
334.
335.  -----
336.  --  INDEX DE LA TABLE STOCKER
337.  -----
338.
339.  CREATE INDEX I_FK_STOCKER_MAGASIN
340.    ON STOCKER (MAGNUM ASC)
341.    ;
342.
343.  CREATE INDEX I_FK_STOCKER_PRODUIT
344.    ON STOCKER (PRODNUM ASC)
345.    ;
346.
347.  -----
348.  --  TABLE : FOURNIR
349.  -----
350.
351.  CREATE TABLE FOURNIR
352.  (
353.    FOURNUM CHAR(32) NOT NULL,
354.    PRODNUM CHAR(32) NOT NULL,
355.    HISTODATE CHAR(32) NOT NULL,
```

```
356.      PROPRIX CHAR(32) NULL,
357.      SEUIL CHAR(32) NULL
358.  , CONSTRAINT PK_FOURNIR PRIMARY KEY (FOURNUM, PRODNUM, HISTODATE)
359.  );
360.
361.  -----
362.  --      INDEX DE LA TABLE FOURNIR
363.  -----
364.
365.  CREATE INDEX I_FK_FOURNIR_FOURNISSEUR
366.      ON FOURNIR (FOURNUM ASC)
367.  ;
368.
369.  CREATE INDEX I_FK_FOURNIR_PRODUIT
370.      ON FOURNIR (PRODNUM ASC)
371.  ;
372.
373.  -----
374.  --      TABLE : CONCERNERGLOB
375.  -----
376.
377.  CREATE TABLE CONCERNERGLOB
378.  (
379.      PRODNUM CHAR(32) NOT NULL,
380.      COMGLOBNUM CHAR(32) NOT NULL,
381.      QTECOMGLOB CHAR(32) NULL
382.  , CONSTRAINT PK_CONCERNERGLOB PRIMARY KEY (PRODNUM, COMGLOBNUM)
383.  );
384.
385.  -----
386.  --      INDEX DE LA TABLE CONCERNERGLOB
387.  -----
388.
389.  CREATE INDEX I_FK_CONCERNERGLOB_PRODUIT
390.      ON CONCERNERGLOB (PRODNUM ASC)
391.  ;
392.
393.  CREATE INDEX I_FK_CONCERNERGLOB_COMMANDEGLO
394.      ON CONCERNERGLOB (COMGLOBNUM ASC)
395.  ;
396.
397.
398.  -----
399.  --      CREATION DES REFERENCES DE TABLE
400.  -----
401.
402.
403.  ALTER TABLE BONLIVRAISON ADD (
404.      CONSTRAINT FK_BONLIVRAISON_COMMANDEGLOBAL
405.      FOREIGN KEY (COMGLOBNUM)
406.      REFERENCES COMMANDEGLOBALE (COMGLOBNUM)) ;
407.
```

```
408. ALTER TABLE COMMANDE ADD (
409.     CONSTRAINT FK_COMMANDE_MAGASIN
410.     FOREIGN KEY (MAGNUM)
411.     REFERENCES MAGASIN (MAGNUM)) ;
412.
413. ALTER TABLE COMMANDEGLOBALE ADD (
414.     CONSTRAINT FK_COMMANDEGLOBALE_FOURNISSEUR
415.     FOREIGN KEY (FOURNUM)
416.     REFERENCES FOURNISSEUR (FOURNUM)) ;
417.
418. ALTER TABLE CAMION ADD (
419.     CONSTRAINT FK_CAMION_MODELE
420.     FOREIGN KEY (MODNUM)
421.     REFERENCES MODELE (MODNUM)) ;
422.
423. ALTER TABLE MAGASIN ADD (
424.     CONSTRAINT FK_MAGASIN_SECTEUR
425.     FOREIGN KEY (SECNUM)
426.     REFERENCES SECTEUR (SECNUM)) ;
427.
428. ALTER TABLE CHARGEMENT ADD (
429.     CONSTRAINT FK_CHARGEMENT_CAMION
430.     FOREIGN KEY (CAMNUM)
431.     REFERENCES CAMION (CAMNUM)) ;
432.
433. ALTER TABLE COLIS ADD (
434.     CONSTRAINT FK_COLIS_CHARGEMENT
435.     FOREIGN KEY (CHARNUM)
436.     REFERENCES CHARGEMENT (CHARNUM)) ;
437.
438. ALTER TABLE COLIS ADD (
439.     CONSTRAINT FK_COLIS_PRODUIT
440.     FOREIGN KEY (PRODNUM)
441.     REFERENCES PRODUIT (PRODNUM)) ;
442.
443. ALTER TABLE COLIS ADD (
444.     CONSTRAINT FK_COLIS_COMMANDE
445.     FOREIGN KEY (COMNUM)
446.     REFERENCES COMMANDE (COMNUM)) ;
447.
448. ALTER TABLE LIVRER ADD (
449.     CONSTRAINT FK_LIVRER_PRODUIT
450.     FOREIGN KEY (PRODNUM)
451.     REFERENCES PRODUIT (PRODNUM)) ;
452.
453. ALTER TABLE LIVRER ADD (
454.     CONSTRAINT FK_LIVRER_BONLIVRAISON
455.     FOREIGN KEY (BONLIVNUM)
456.     REFERENCES BONLIVRAISON (BONLIVNUM)) ;
457.
458. ALTER TABLE CONCERNER ADD (
459.     CONSTRAINT FK_CONCERNER_COMMANDE
```

```
460.      FOREIGN KEY (COMNUM)
461.      REFERENCES COMMANDE (COMNUM)) ;
462.
463.      ALTER TABLE CONCERNER ADD (
464.      CONSTRAINT FK_CONCERNER_PRODUIT
465.      FOREIGN KEY (PRODNUM)
466.      REFERENCES PRODUIT (PRODNUM)) ;
467.
468.      ALTER TABLE ASSOCIER ADD (
469.      CONSTRAINT FK_ASSOCIER_COMMANDE
470.      FOREIGN KEY (COMNUM)
471.      REFERENCES COMMANDE (COMNUM)) ;
472.
473.      ALTER TABLE ASSOCIER ADD (
474.      CONSTRAINT FK_ASSOCIER_COMMANDEGLOBALE
475.      FOREIGN KEY (COMGLOBNUM)
476.      REFERENCES COMMANDEGLOBALE (COMGLOBNUM)) ;
477.
478.      ALTER TABLE ASSOCIER ADD (
479.      CONSTRAINT FK_ASSOCIER_PRODUIT
480.      FOREIGN KEY (PRODNUM)
481.      REFERENCES PRODUIT (PRODNUM)) ;
482.
483.      ALTER TABLE STOCKER ADD (
484.      CONSTRAINT FK_STOCKER_MAGASIN
485.      FOREIGN KEY (MAGNUM)
486.      REFERENCES MAGASIN (MAGNUM)) ;
487.
488.      ALTER TABLE STOCKER ADD (
489.      CONSTRAINT FK_STOCKER_PRODUIT
490.      FOREIGN KEY (PRODNUM)
491.      REFERENCES PRODUIT (PRODNUM)) ;
492.
493.      ALTER TABLE FOURNIR ADD (
494.      CONSTRAINT FK_FOURNIR_FOURNISSEUR
495.      FOREIGN KEY (FOURNUM)
496.      REFERENCES FOURNISSEUR (FOURNUM)) ;
497.
498.      ALTER TABLE FOURNIR ADD (
499.      CONSTRAINT FK_FOURNIR_PRODUIT
500.      FOREIGN KEY (PRODNUM)
501.      REFERENCES PRODUIT (PRODNUM)) ;
502.
503.      ALTER TABLE CONCERNERGLOB ADD (
504.      CONSTRAINT FK_CONCERNERGLOB_PRODUIT
505.      FOREIGN KEY (PRODNUM)
506.      REFERENCES PRODUIT (PRODNUM)) ;
507.
508.      ALTER TABLE CONCERNERGLOB ADD (
509.      CONSTRAINT FK_CONCERNERGLOB_COMMANDEGLOBA
510.      FOREIGN KEY (COMGLOBNUM)
511.      REFERENCES COMMANDEGLOBALE (COMGLOBNUM)) ;
```

```
512.  
513.  
514. -----  
515. --      FIN DE GENERATION  
516. -----
```

### 1-2-2 : Création des utilisateurs

Nous avons créé deux utilisateurs, le chef de projet qui gère le système, et le fournisseur 4 qui peut consulter et chercher que les données concernant lui-même.

Pour mieux contrôler le droit d'utilisation de la base de données, nous devons limiter le privilège de l'utilisateur fournisseur 4.

D'abord le fournisseur peut consulter les commandes globales concernant lui-même avec les produits correspondants. Quand il commence à préparer des produits d'une commande globale, il peut modifier que l'état de cette commande globale. Pour que le fournisseur peut consulter les produits en attente de livrer, nous lui donnons le droit de consulter les données de bon de livraison. C'est le fournisseur qui indique le numéro de bon de livraison, donc il a le privilège d'insérer des données dans le table **Bonlivraison**. Grâce au droit de modifier le propre prix du jour et le seuil d'approvisionnement, nous donnons le droit de consulter et mettre à jour sur le fournir sur le prix et seuil.

Deuxièmement, nous donnons le droit de création des view, de procédure, et de déclencheur au fournisseur, afin de simplifier et visualiser des consultation et modification des données.

```
1. GRANT SELECT ON COMMANDEGLOBALE TO "AEROFRANCE_LIAO" ;  
2. GRANT UPDATE(COMGLOBETAT) ON COMMANDEGLOBALE TO "AEROFRANCE_LIAO" ;  
3. GRANT SELECT ON LIVRER TO "AEROFRANCE_LIAO" ;  
4. GRANT SELECT ON BONLIVRAISON TO "AEROFRANCE_LIAO" ;  
5. GRANT INSERT ON BONLIVRAISON TO "AEROFRANCE_LIAO" ;  
6. GRANT SELECT ON CONCERNERGLOB TO "AEROFRANCE_LIAO" ;  
7. GRANT SELECT ON FOURNIR TO "AEROFRANCE_LIAO" ;  
8. GRANT UPDATE(proprix, seuil) ON FOURNIR TO "AEROFRANCE_LIAO" ;  
9. GRANT SELECT ON view_bonliv TO "AEROFRANCE_LIAO" ;  
10.  
11. GRANT create view to "AEROFRANCE_LIAO" ;  
12. GRANT create TRIGGER to "AEROFRANCE_LIAO" ;  
13. GRANT create PROCEDURE to "AEROFRANCE_LIAO" ;
```

Le fournisseur est un personnel de fournisseur 4, d'après cette hypothèse, nous avons créé des procédures, des views et des déclencheurs pour cet utilisateur.

#### VIEWS:

##### VIEW\_Bonlivraison

Visualisation des bons de livraison créés par le fournisseur avec la quantité reçue et refusée par la centrale d'achat.

```
1. CREATE OR REPLACE FORCE VIEW "AEROFRANCE_LIAO"."VIEW_BONLIVRAISON" ("BONLIVNUM", "PRODNUM", "QTELIV",  
   "QTEREFUS") AS  
2. SELECT bonlivnum, prodnum, qteliv, qterefus  
3. FROM CHEF_GUO.VIEW_BONLIV  
4. WHERE fournum = 4;
```

##### VIEW\_COMMANDEGLOBALE

Visualisation des commandes globales concernant ce fournisseur. Le fournisseur peut modifier l'état de commande globale directement sur ce table.

```
1. CREATE OR REPLACE FORCE VIEW "AEROFRANCE_LIAO"."VIEW_COMMANDEGLOBALE" ("COMGLOBNUM",  
"FOURNUM", "COMGLOBDATE", "COMGLOBETAT") AS  
2. SELECT "COMGLOBNUM", "FOURNUM", "COMGLOBDATE", "COMGLOBETAT"  
3. FROM CHEF_GUO.COMMANDEGLOBALE  
4. WHERE FOURNUM = 4;
```

### VIEW\_FOURNIR

Visualisation des prix et seuils historiques proposés par fournisseur 4. Le fournisseur peut modifier directement le propre prix et le seuil directement sur ce table.

```
1. CREATE OR REPLACE FORCE VIEW "AEROFRANCE_LIAO"."VIEW_FOURNIR" ("PRODNUM", "HISTODATE", "PROPRIX",  
"SEUIL") AS  
2. SELECT "PRODNUM", "HISTODATE", "PROPRIX", "SEUIL"  
3. FROM CHEF_GUO.FOURNIR  
4. WHERE FOURNUM = 4;
```

### VIEW\_PRODUIT

Visualisation des produits en attente de livrer avec la commande globale concernée et la quantité commandée.

```
1. CREATE OR REPLACE FORCE VIEW "AEROFRANCE_LIAO"."VIEW_PRODUIT" ("COMGLOBNUM", "PRODNUM",  
"QTECOMGLOB") AS  
2. select co.comglobnum, co.prodnum, co.qtecomglob  
3. from CHEF_GUO.concernerglob co, CHEF_GUO.commandeglobale com  
4. where com.comglobnum = co.comglobnum  
5. and com.fournum = 4  
6. and co.prodnum not in (select l.prodnum  
7. from CHEF_GUO.livrer l, CHEF_GUO.bonlivraison b, CHEF_GUO.concernerglob c  
8. where l.bonlivnum = b.bonlivnum  
9. and com.comglobnum = b.comglobnum  
10. and com.comglobnum = c.comglobnum  
11. and c.prodnum = l.prodnum  
12. and com.fournum = 4  
13. and l.qteliv-l.qterefus = c.qtecomglob );
```



### Procédures :

Les deux procédures au-dessous permettent la transaction autonome sur le table fournir et préparation.

#### MODIFICATION\_FOURNIR

```
1. create or replace PROCEDURE MODIFICATION_FOURNIR
2. (
3.   P_PRODNUM IN NUMBER,
4.   P_DATE IN DATE,
5.   P_PRIX IN NUMBER,
6.   P_SEUIL IN NUMBER
7. ) AS pragma AUTONOMOUS_TRANSACTION;
8. BEGIN
9.   update CHEF_GUO.fournir SET PROPRIX = P_PRIX, SEUIL = P_SEUIL
10.  where FOURNUM = 4
11.    and HISTODATE = P_DATE
12.    and PRODNUM = P_PRODNUM;
13.  commit;
14. END MODIFICATION_FOURNIR;
```

#### ETAT\_PREPARATION

```
1. create or replace PROCEDURE ETAT_PREPARATION
2. (
3.   P_COMGLOBNUM IN NUMBER
4. ) AS
5.   pragma AUTONOMOUS_TRANSACTION;
6. BEGIN
7.   update CHEF_GUO.commandeglobale SET comglobetat = 'en cours de préparation' where comglobnum =
   P_COMGLOBNUM;
8.   commit;
9. END ETAT_PREPARATION;
```

### Déclencheurs :

Ces deux déclencheurs permettent de limiter la modification des données.

Le fournisseur ne peut que modifier le propre prix et le seuil de la date du jour et l'état de commande global en 'en cours de préparation'.

#### MODIFIER\_FOURNIR

```
1. create or replace TRIGGER MODIFIER_FOURNIR
2. INSTEAD OF UPDATE ON VIEW_FOURNIR
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.     if to_char(:n.histodate, 'DD/MM/YY') = to_char(sysdate, 'DD/MM/YY') then
7.         modification_fournir(:n.prodnum, :n.histodate, :n.proprix, :n.seuil);
8.     end if;
9. END;
```

#### PREPARATION\_COMGLOB

```
1. create or replace TRIGGER PREPARATION_COMGLOB
2. INSTEAD OF UPDATE ON VIEW_COMMANDEGLOBALE
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.     if :n.comglobetat = 'en cours de préparation' then
7.         ETAT_PREPARATION(:n.comglobnum);
8.     end if;
9. END;
```

Enfin, nous avons révoqué le droit de création de view, de procédure et de déclencheur du fournisseur afin de mieux gérer le privilège de fournisseur sur la base de données.

```
14. revoke create view from "AEROFRANCE_LIAO" ;
15. revoke create TRIGGER from "AEROFRANCE_LIAO" ;
16. revoke create PROCEDURE from "AEROFRANCE_LIAO" ;
17. revoke INSERT ON view_bonliv FROM "AEROFRANCE_LIAO" ;
```

## Partie II : Rédaction d'un dossier de spécifications fonctionnelles

### 2-1 : Hypothèses et règles de gestion concernant la dynamique

#### 2-1-1 : Hypothèses

**BonLivraison :**

-- Le fournisseur crée des bons de livraison par lui-même, avec bonlivnum et comglobnum.

**Livrer :**

-- Quand la centrale reçoit la livraison, elle remplit la table de 'LIVRER', avec prodnum, qteliv (qterefus est calculé automatiquement par des procédures selon des conditions).

**Fournir :**

-- Chaque jour il faut un prix et un seuil sur un produit d'un fournisseur : le système récupère le prix et le seuil du jour précédent automatiquement, et s'il y a des changements, le fournisseur peut les modifier.

**Chargement :**

-- Au début de chaque jour, nous allons mettre à jour des états des chargements.

-- Chaque jour nous créons des chargements en fonction de secteur manuellement, nous cherchons des colis en attente de prise en charge et les met dans le chargement avec le même secteur.

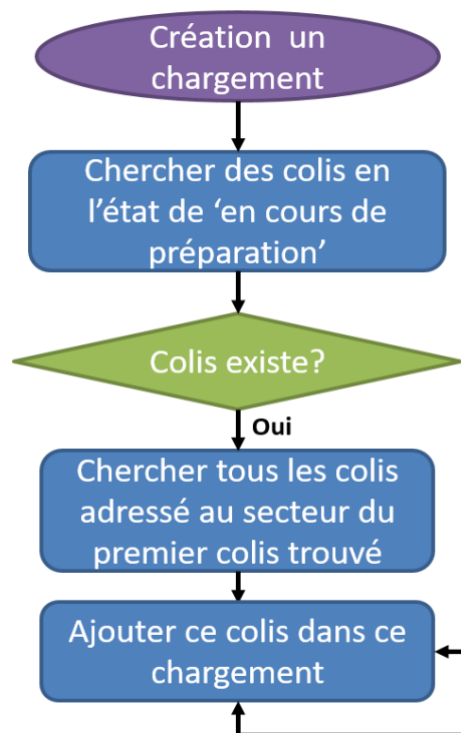
-- Chaque jour un camion est affecté sur un seul chargement.

#### 2-1-2 : Règles de gestion concernant la dynamique

Règles	Procédures	Fonctions	Déclencheurs	Séquences
Création des chargements	CREATION_CHARGEMENT;		CHARGEMENTNUM_AUTO;	SEQ_CHARGEMENT;
Création des commandes	RG_PREPARATION_COMMANDE;	RG_QTE_COMMANDE;	CREATION_COMMANDE; INSERER_LIGNE_COMMANDE;	SEQ_COMMANDE;
Factorisation des commandes	RG_FACTORISATION_COMMANDE;		INSERER_LIGNE_COMGLOB; FACTORISATION_COMMANDE;	SEQ_COMGLOB;
Création des bonlivraison	(par fournisseur)		BONLIVENUM_AUTO	SEQ_BONLIV
Contrôle de la quantité de produit livré	RG_PROD_QTE_LIV;		CONTRÔLE_QTE_BONLIV	
Distribution des colis	RG_DISTRIBUTION_COLIS; COLIS_LIVRE;		LIVRAISON_COLIS; EXPÉDIER_CHARGEMENT;	SEQ_COLIS

### 2-1-2-1 : Création des chargements

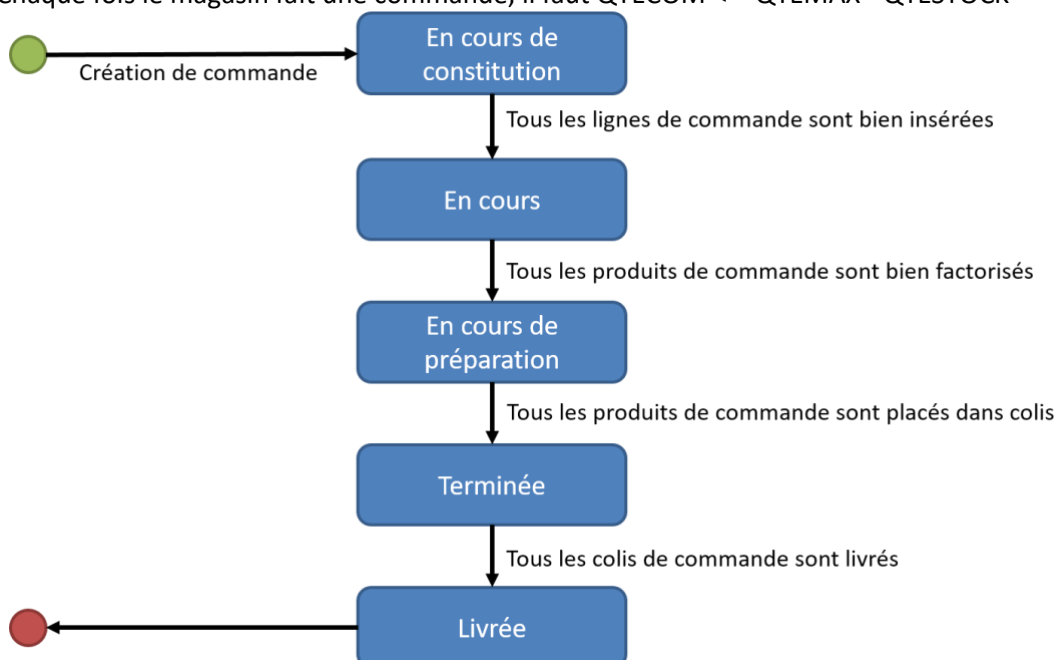
Au début de chaque jour, le système crée 4 chargements vides pour 4 secteurs (état initial : 'en cours'), et il cherche des colis précédents qui sont en attente de prise en charge, et les met dans les chargements. Le secteur de chargement s'adresse au secteur du premier colis dans ce chargement.



### 2-1-2-2 : Création des commandes

Lors de la création d'une commande, son état initial est 'en cours de constitution', et ensuite nous ajoutons des produits de cette commande dans la table 'Concerner'. Quand tous les produits concernés sont bien y ajoutés, l'état de commande passe en 'en cours', et nous ne peut plus modifier cette commande.

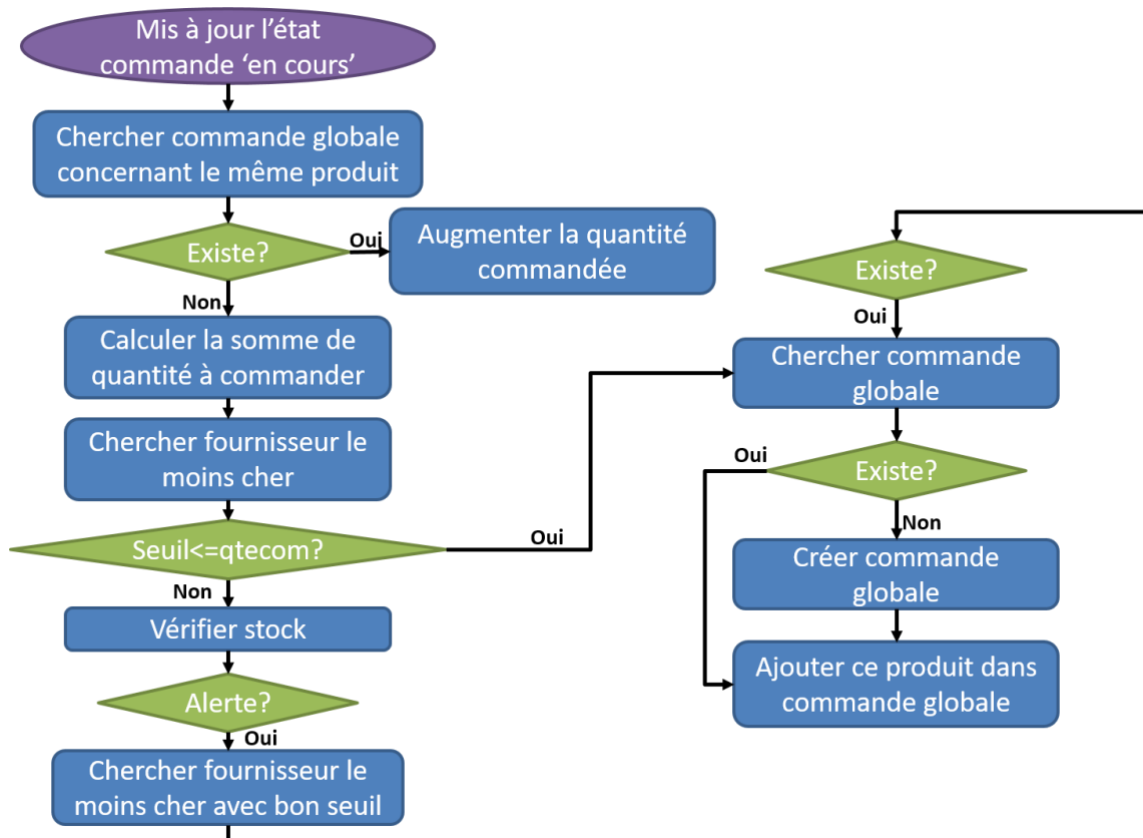
Chaque fois le magasin fait une commande, il faut  $QTECOM \leq QTEMAX - QTESTOCK$



### 2-1-2-3 : Factorisation des commandes

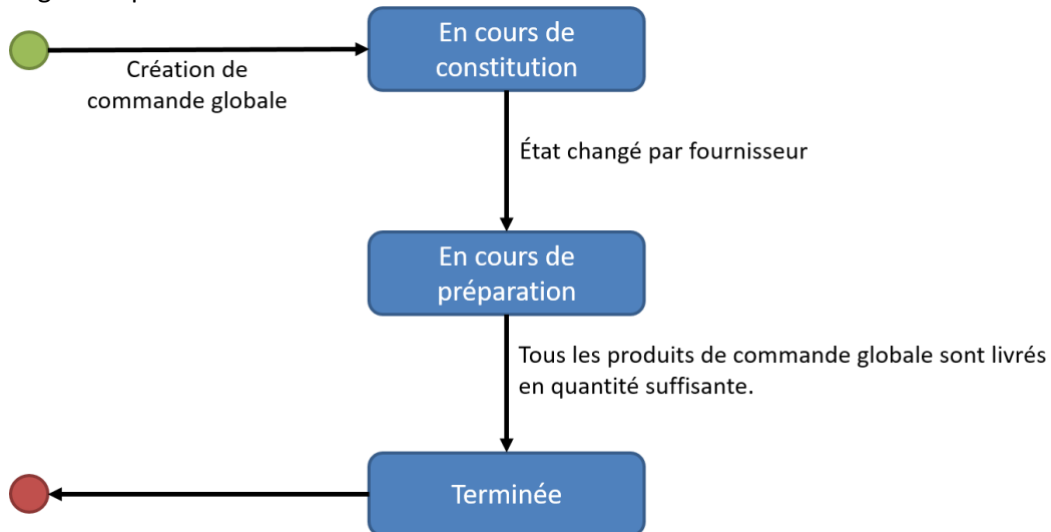
Après avoir complété la commande, l'état passe en 'en cours' et nous pouvons donc commencer à répartir la commande. Le système va factoriser tous les produits en fonction des prix et seuils offerts par les fournisseurs, l'état de stock de magasin et l'état de commande globale existe.

D'abord le système cherche les commandes globales 'en cours de constitution' concernant le même produit, s'il existe, le produit à commander sera ajouté directement dans cette commande globale en augmentant la quantité. S'il n'existe pas une commande globale que nous cherchons, le système cherche tous les produit en attente de factorisation et calcule la somme de quantité à commander. Après, le système cherche le fournisseur le moins cher et vérifie le seuil. Si le seuil proposé est inférieur ou égal à la quantité à commander, le système cherche une commande globale de ce fournisseur, soit nous ajoutons une ligne dans la commande globale existe, soit nous créons une nouvelle commande globale s'il n'existe pas de commande globale de ce fournisseur. Si le seuil est supérieur à la quantité à commander, le système vérifie si le produit est en alerte de stock. Si c'est le cas en alerte, le système cherche le fournisseur le moins cher concernant le même produit et proposant le seuil inférieur ou égal à la quantité à commander. S'il existe, le système cherche une commande globale de ce fournisseur, soit nous ajoutons une ligne dans la commande globale existe, soit nous créons une nouvelle commande globale s'il n'existe pas de commande globale de ce fournisseur. Si nous ne trouvons pas, le produit doit attendre des nouvelles commandes pour atteindre le seuil.



#### 2-1-2-4 : Préparation de commande globale et création des bons livraison.

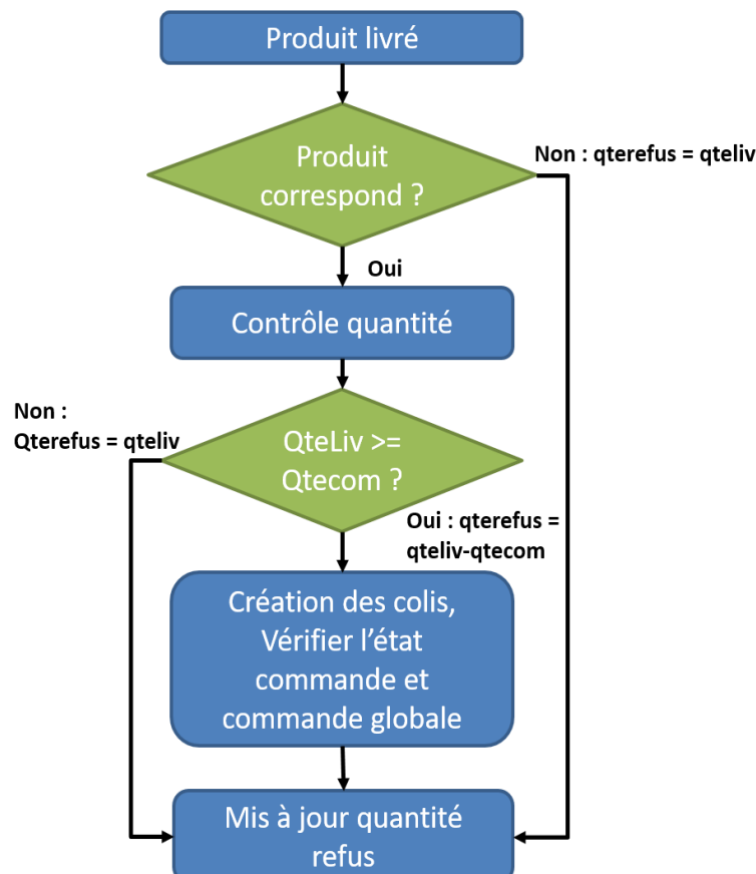
Quand le fournisseur commence à préparer les produits d'une commande, il modifie l'état de la commande globale concernée en 'en cours de préparation'. Il crée un bon de livraison quand il livre des produits. Quand tous les produits d'une commande globale sont livrés avec la bonne quantité, l'état de cette commande globale passe en 'terminée'.



#### 2-1-2-5 : Contrôle de la quantité de produit livré

Après le fournisseur faire des bons livraison, nous allons reçu des produits concernés et contrôler la quantité livrée selon le produit commandé et la quantité commandée.

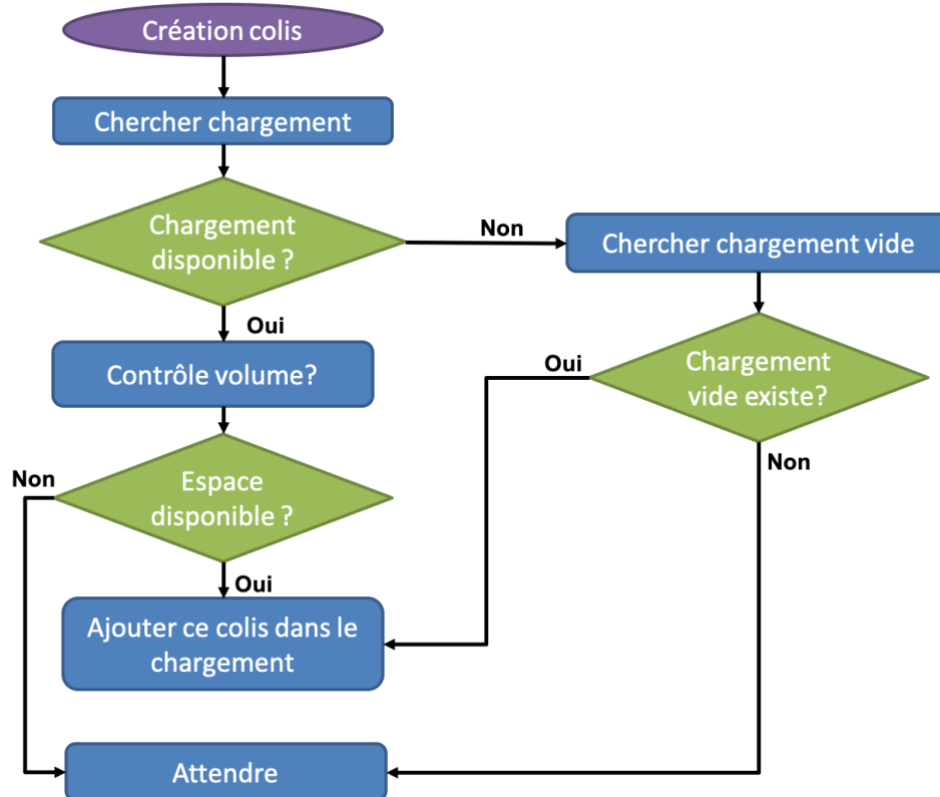
Si le produit n'est pas concerné, le système refuse tous les produits. Si le produit correspondant, mais la quantité est inférieure à la quantité commandée ou le produit a été déjà bien reçu avec la bonne quantité, tous les produits seront refusés. Si la quantité est supérieure à la quantité commandée, le système refuse la quantité excédante.



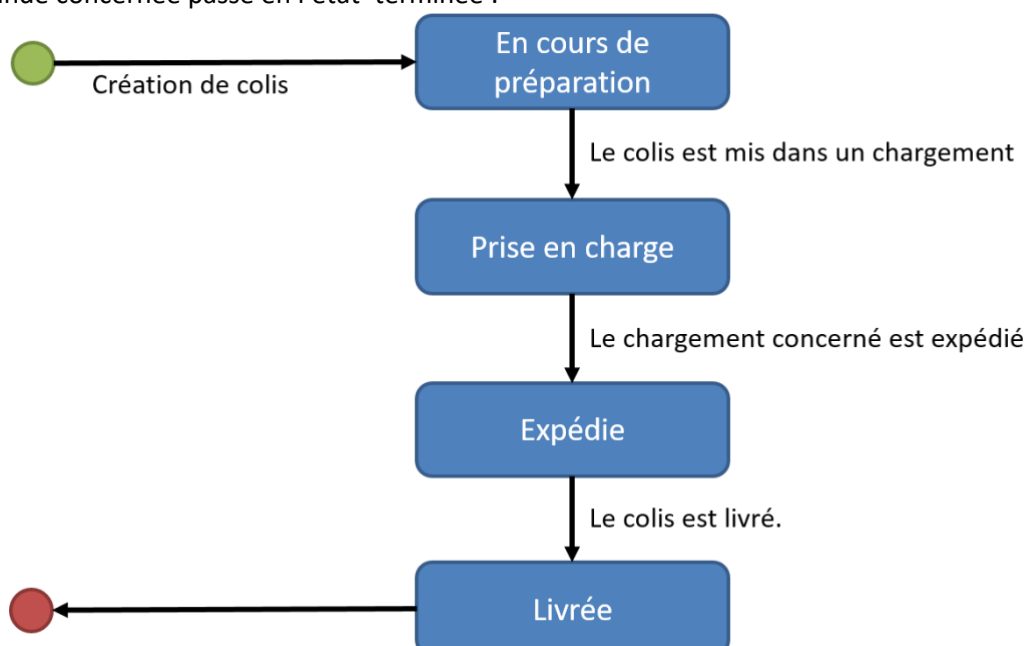
### 2-1-2-6 : Distribution des colis

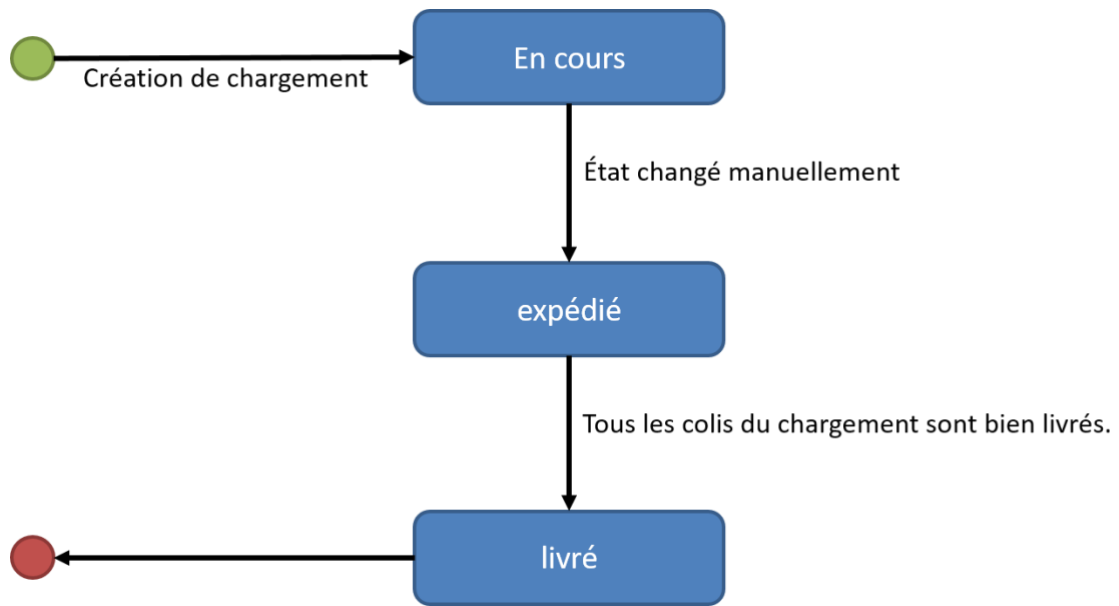
Dès la création d'un colis (état initial : 'en cours de préparation'), le système va distribuer le colis dans chargement en fonction de secteur et l'espace disponible (la capacité du modèle de camion affecté au chargement moins la somme des volumes des colis qui sont déjà dans ce chargement) :

$$\text{Le volume d'un colis} = \text{ProdVolume} * \text{QteLiv}$$



Quand nous ajoutons ce colis dans un chargement, l'état de ce colis passe en 'prise en charge', et quand nous expédions un chargement, tous les colis dans ce chargement passent en l'état 'expédié' en même temps. Quand tous les colis concernés sont bien livrés, le chargement pass en l'état 'livrée' et la commande concernée passe en l'état 'terminée'.







## 2-2 : Décomposition modulaire

### 2-2-1 : Procédures

#### 2-2-1-1 : RG\_PREPARATION\_COMMANDE

Cette procédure est pour vérifier qu'est-ce qu'il reste des produits d'une commande ne sont pas encore factorisés, si non, l'état de cette commande passe en 'en cours de préparation'.

```
1. create or replace PROCEDURE RG_PREPARATION_COMMANDE
2. (
3.   P_COMNUM IN NUMBER,
4.   P_PRODNUM IN NUMBER
5. ) AS
6.   v_prodnum number;
7.   cursor c_produit is select prodnum
8.     from concerner c
9.    where c.comnum = P_COMNUM
10.    and c.prodnum <> p_prodnum
11.    and c.prodnum not in (select asso.prodnum
12.      from associer asso
13.     where asso.comnum = P_COMNUM);
14. BEGIN
15.   open c_produit;
16.   fetch c_produit into v_prodnum;
17.   if c_produit%notfound then
18.     update commande set cometat = 'en cours de préparation' where comnum = P_COMNUM;
19.     SYS.DBMS_OUTPUT.PUT_LINE('Tous les produit de la commande ' || p_comnum || ' sont factorisé. L"état de comman
20. de passe en "en cours de préparation".');
21.   end if;
22.   close c_produit;
23. END RG_PREPARATION_COMMANDE;
```

#### 2-2-1-2 : RG\_FACTORISATION\_COMMANDE

Cette procédure est le processus entier de répartir des produits d'une commande dans des commandes différentes en fonctions de fournisseur et le stock.

```
1. create or replace PROCEDURE RG_FACTORISATION_COMMANDE
2. (
3.   P_COMNUM IN NUMBER
4. ) AS
5.   v_comglobnum NUMBER;
6.   v_qtecomglob NUMBER;
7.   v_qtestock NUMBER;
8.   v_qtealerte NUMBER;
9.   v_prodnum NUMBER;
10.  v_qteattente NUMBER;
11.  v_fournum NUMBER;
12.  v_seuil NUMBER;
13.  v_inserer BOOLEAN;
14.  cursor c_produit is select prodnum, qtecom
15.    from concerner
16.   where comnum = P_COMNUM;
17.  cursor c_prodattente is select co.comnum, co.qtecom
18.    from commande com, concerner co
19.   where com.comnum = co.comnum
20.   and com.comnum <> p_comnum
21.   and prodnum = v_prodnum
22.   and com.cometat = 'en cours'
23.   and prodnum not in (select asso.prodnum
24.     from associer asso
25.    where asso.comnum = com.comnum);
26.  cursor c_fournisseur is select f1.fournum, f1.seuil
27.    from fournir f1
28.   where f1.prodnum = v_prodnum
29.   and to_char(f1.histodate, 'DD/MM/YY') = to_char(sysdate, 'DD/MM/YY')
30.   and f1.proprix = (select min(proprix)
```

```

31.         from fournisseur f2
32.         where f2.prodnum = v_prodnum
33.         and to_char(f2.histodate, 'DD/MM/YY') = to_char(sysdate, 'DD/MM/YY'));
34. cursor c_comglob_produit is select comglob.comglobnum
35.         from CONCERNERGLOB co, commandeglobale comglob
36.         where comglob.COMGLOBNUM = co.comglobnum
37.         and co.PRODNUM = v_prodnum
38.         and comglob.COMGLOBETAT = 'en cours de constitution';
39. cursor c_comglob is select comglobnum
40.         from COMMANDEGLOBALE
41.         where comglobetat = 'en cours de constitution'
42.         and fournum = v_fournum;
43. cursor c_four_alerte is select f1.fournum
44.         from fournisseur f1
45.         where f1.prodnum = v_prodnum
46.         and f1.seuil <= v_qtecomglob
47.         and to_char(f1.histodate, 'DD/MM/YY') = to_char(sysdate, 'DD/MM/YY')
48.         and f1.proprix = (select min(proprix)
49.         from fournisseur f2
50.         where f2.prodnum = v_prodnum
51.         and f2.seuil <= v_qtecomglob
52.         and to_char(f2.histodate, 'DD/MM/YY') = to_char(sysdate, 'DD/MM/YY'));
53. BEGIN
54.   SYS.DBMS_OUTPUT.PUT_LINE('Commande ' || p_comnum);
55.   for un_produit in c_produit loop
56.     SYS.DBMS_OUTPUT.PUT_LINE('Produit' || un_produit.prodnum);
57.     v_prodnum := un_produit.prodnum;
58.     v_inserer := false;
59.     --chercher une commande globale concernée à ce produit
60.     open c_comglob_produit;
61.     fetch c_comglob_produit into v_comglobnum;
62.     if c_comglob_produit%found then
63.       --s'il existe une commande globale concerné
64.       update concernerglob set qtecomglob = qtecomglob + un_produit.qtecom
65.       where comglobnum = v_comglobnum and prodnum = un_produit.prodnum;
66.       insert into associer values(p_comnum, v_comglobnum, un_produit.prodnum);
67.       SYS.DBMS_OUTPUT.PUT_LINE('produit ' || un_produit.prodnum || ' --
68.       existe une commande globale, augmenter la qtecomglob');
69.     else
70.       close c_comglob_produit;
71.       --sinon, calculer la somme de la quantité à commander
72.       v_qtecomglob := un_produit.qtecom;
73.       for un_prodattente in c_prodattente loop
74.         v_qtecomglob := v_qtecomglob + un_prodattente.qtecom;
75.       end loop;
76.       --chercher le fournisseur le moins cher concernant le même produit
77.       open c_fournisseur;
78.       fetch c_fournisseur into v_fournum, v_seuil;
79.       while c_fournisseur%found and v_inserer = false loop
80.         SYS.DBMS_OUTPUT.PUT_LINE('Le fournisseur ' || v_fournum || ' est le moins cher');
81.         --s'il existe, vérifier le seuil
82.         if v_seuil <= v_qtecomglob then
83.           SYS.DBMS_OUTPUT.PUT_LINE('On atteint le seuil');
84.           --si le seuil est inférieur ou égale à la quantité à commander, chercher la commande globale de ce fournisseur
85.           open c_comglob;
86.           fetch c_comglob into v_comglobnum;
87.           if c_comglob%notfound then
88.             --s'il n'existe pas de commande globale, créer une commande globale
89.             insert into commandeglobale values(seq_comglob.nextval, v_fournum, sysdate, 'en cours de constitution');
90.             v_comglobnum := seq_comglob.currval;
91.             SYS.DBMS_OUTPUT.PUT_LINE('Commande n'existe pas, créer une commande globale');
92.           end if;

```

```

93.     close c_comglob;
94.     --insérer une ligne dans commandeglobale avec la somme des quantité
95.     insert into concernerglob values (un_produit.prodnum,v_comglobnum,v_qtecomglob);
96.     SYS.DBMS_OUTPUT.PUT_LINE('Insérer une ligne de commande globale du produit' || un_produit.prodnum);
97.     --insérer les lignes dans le table associer
98.     insert into associer values(P_comnum, v_comglobnum, un_produit.prodnum);
99.     for un_prodattente in c_prodattente loop
100.        insert into associer values(un_prodattente.comnum, v_comglobnum, un_produit.prodnum);
101.     end loop;
102.     SYS.DBMS_OUTPUT.PUT_LINE('Insérer tous les commande concerné du produit ' || un_produit.prodnum || ' dans le
table associer');
103.     v_inserer := true;
104.     else
105.        fetch c_fournisseur into v_fournum, v_seuil;
106.     end if;
107. end loop;
108. close c_fournisseur;
109. if v_inserer = false then
110.    SYS.DBMS_OUTPUT.PUT_LINE('On n''atteint pas le seuil');
111.    --vérifier cas alerte
112.    select qtestock, qtealerte into v_qtestock, v_qtealerte
113.    from commande c, stocker s
114.    where c.comnum = P_COMNUM
115.    and c.magnum = s.magnum
116.    and s.prodnum = un_produit.prodnum;
117.    if v_qtestock <= v_qtealerte then
118.        SYS.DBMS_OUTPUT.PUT_LINE('stock alerte du produit ' || un_produit.prodnum);
119.        --cas stock alerte
120.        chercher le fournisseur le moins cher concernant le même produit avec un seuil inférieur à la quantité totale à commander
121.        open c_four_alerte;
122.        fetch c_four_alerte into v_fournum;
123.        if c_four_alerte%found then
124.            --s'il existe, chercher la commande globale de ce fournisseur
125.            SYS.DBMS_OUTPUT.PUT_LINE('Le fournisseur ' || v_fournum || ' est le moins cher avec bon seuil du produit ' || un
_produit.prodnum);
126.            open c_comglob;
127.            fetch c_comglob into v_comglobnum;
128.            if c_comglob%notfound then
129.                --s'il n'existe pas de commande globale, créer une commande globale
130.                insert into commandeglobale values(seq_comglob.nextval,v_fournum,sysdate,'en cours de constitution');
131.                v_comglobnum := seq_comglob.currval;
132.                SYS.DBMS_OUTPUT.PUT_LINE('Commande n''existe pas, créer une commande globale');
133.            end if;
134.            --insérer une ligne dans commandeglobale avec la somme des quantité
135.            insert into concernerglob values (un_produit.prodnum,v_comglobnum,v_qtecomglob);
136.            SYS.DBMS_OUTPUT.PUT_LINE('Insérer une ligne de commande globale du produit ' || un_produit.prodnum);
137.            --insérer les lignes dans le table associer
138.            insert into associer values(P_comnum, v_comglobnum, un_produit.prodnum);
139.            for un_prodattente in c_prodattente loop
140.                insert into associer values(un_prodattente.comnum, v_comglobnum, un_produit.prodnum);
141.            end loop;
142.            SYS.DBMS_OUTPUT.PUT_LINE('Insérer tous les commande concerné du produit ' || un_produit.prodnum || ' dans l
e table associer');
143.            close c_comglob;
144.        end if;
145.        close c_four_alerte;
146.    end if;
147. end if;
148. end if;
149. end loop;
150. SYS.DBMS_OUTPUT.PUT_LINE('Factorisation de la commande ' || p_comnum || ' est finie');
151. END RG_FACTORISATION_COMMANDE;

```

### 2-2-1-3 : RG\_TERMINER\_COMMANDE

C'est pour vérifier qu'est-ce que tous les produits d'une commande sont bien placés dans colis, si oui, l'état de cette commande passe en 'terminée'.

```
1. create or replace PROCEDURE RG_TERMINER_COMMANDE
2. (
3.   p_comnum IN NUMBER,
4.   p_prodnum IN NUMBER
5. )
6. AS
7.   v_produit NUMBER;
8.   cursor c_produit is select prodnum from concerner c
9.   where c.comnum = p_comnum and c.prodnum <> p_prodnum and prodnum not in (select prodnum from colis co
10.                                     where co.comnum = p_comnum and co.prodnum <> p_prodnum);
11. BEGIN
12.   open c_produit;
13.   fetch c_produit into v_produit;
14.   if c_produit%notfound then
15.     SYS.DBMS_OUTPUT.PUT_LINE('Tous les produit de la commande ' || p_prodnum || ' sont placés dans colis');
16.   update commande set cometat = 'terminée'
17.   where comnum = p_comnum;
18.   end if;
19.   close c_produit;
20. END RG_TERMINER_COMMANDE;
```

### 2-2-1-4 : RG\_PROD\_QTE\_LIV

C'est la contrôle de la quantité livrée d'un produit de bon de bon livraison, si le produit est livré en quantité suffisant et il est livré qu'une seul fois sur un seul bon livraison, le système va commencer de faire des colis. Le système va mettre à jour la quantité de refus dans n'importe quelle condition.

```
1. create or replace PROCEDURE RG_PROD_QTE_LIV
2. (
3.   p_prodnum IN NUMBER
4. , p_bonlivnum IN NUMBER
5. , p_qteliv IN NUMBER
6. ) AS
7.   v_qterefus NUMBER;
8.   v_comglobnum NUMBER;
9.   v_qtecomglob NUMBER;
10.  v_comnum NUMBER;
11.  v_prodvolum NUMBER;
12.  v_qtecom NUMBER;
13.  v_colvolum NUMBER;
14.  v_autrebbonlivnum NUMBER;
15.  v_prodatte NUMBER;
16.  v_bonlivnum NUMBER;
17.  v_prodlivre NUMBER;
18.  v_prodconcerne NUMBER;
19.  v_colnum NUMBER;
20.  -- trouver la quantité de ce produit dans cette commande globale
21.  cursor c_qtecomglob is select qtecomglob from concernerglob cg
22.  where cg.COMGLOBNUM = v_comglobnum and cg.PRODNUM=p_prodnum;
23.  -- trouver commande associé avec ce produit et avec cette commande globale pour faire colis
24.  cursor c_commande is select comnum from associer a
25.  where a.COMGLOBNUM = v_comglobnum and a.prodnum = p_prodnum;
26.  - trouver l'existence de bon livraison qui est déjà reçu avant sur ce produit et sur cette commande globale.
27.  cursor c_autrebbonlivraison is select bon.bonlivnum
28.  from livrer l, bonlivraison bon
29.  where bon.comglobnum = v_comglobnum
30.  and l.bonlivnum = bon.bonlivnum
31.  and l.prodnum = p_prodnum
32.  and l.bonlivnum <> p_bonlivnum
33.  and l.qteliv - l.qterefus = v_qtecomglob;
34. BEGIN
35.   SYS.DBMS_OUTPUT.PUT_LINE('Bonlivraison ' || p_bonlivnum || ' : produit ' || p_prodnum || ' quantité : ' || p_qteliv);
```

```

36. -- trouver la commande globale concerné de ce produit et de ce bon livraison
37. select comglobnum into v_comglobnum from bonlivraison b
38. where b.BONLIVNUM = p_BONLIVNUM;
39. open c_qtecomglob;
40. fetch c_qtecomglob into v_qtecomglob;
41. if c_qtecomglob%notfound then
42. SYS.DBMS_OUTPUT.PUT_LINE('produit ne correspond pas');
43. -- Cas 1 : le produit ne correspond pas à la commande globale
44. v_qterefus := p_QTELIV;
45. --insert into bonlivraison values (seq_bonliv.nextval,v_comglobnum,sysdate);
46. --insert into livrer values (p_prodnun,v_bonlivnum,null,null);
47. --raise_application_error(-20101,'le produit ne correspond pas');
48. else
49. open c_autrebonlivraison;
50. fetch c_autrebonlivraison into v_autrebonlivnum;
51. SYS.DBMS_OUTPUT.PUT_LINE(v_autrebonlivnum);
52. if c_autrebonlivraison%notfound then
53. if p_QTELIV < v_qtecomglob then
54. SYS.DBMS_OUTPUT.PUT_LINE('quantité insuffisant');
55. -- Cas 2 : le produit est livré en quantité insuffisant
56. v_qterefus := p_QTELIV;
57. --insert into bonlivraison values (seq_bonliv.nextval,v_comglobnum,sysdate);
58. --insert into livrer values (p_prodnun,v_bonlivnum,null,null);
59. --raise_application_error(-20101,'quantité insuffisant');
60. elsif p_QTELIV >= v_qtecomglob then
61. SYS.DBMS_OUTPUT.PUT_LINE('quantité dépassant ou bon');
62. -
    Cas 3&4 : le produit est livré en bon quantité ou dans une quantité dépassant la quantité commandée, nous pouvons com
    mencer de faire des colis
63. v_qterefus := p_QTELIV - v_qtecomglob;
64. select count(prodnum) into v_prodconcerne
65. from concernerglob
66. where comglobnum = v_comglobnum
67. and prodnum <> p_prodnun;
68. select count(l.prodnum) into v_prodlivre
69. from livrer l, bonlivraison bon, concernerglob co
70. where bon.comglobnum = co.comglobnum
71. and co.PRODNUM = l.prodnum
72. and bon.bonlivnum = l.bonlivnum
73. and co.comglobnum = v_comglobnum
74. and l.qteliv - l.QTEREFUS = co.qtecomglob
75. and l.prodnum <> p_prodnun;
76. if v_prodlivre = v_prodconcerne then
77. -
    si tous les produit dans cette commande globale sont déjà bien livré, nous pouvons changer l'état de cette commande glob
    ale à 'terminé', sinon, nous faisons rien.
78. SYS.DBMS_OUTPUT.PUT_LINE('la commande globale ' || v_comglobnum || ' est terminée');
79. update commandeglobale set comglobetat = 'terminée' where comglobnum = v_comglobnum;
80. -- faire des colis des commandes sur ce produit et cette commande globale
81. end if;
82. for un_commande in c_commande loop
83. select qtecom into v_qtecom from concerner c
84. where c.COMNUM = un_commande.comnum and c.PRODNUM = p_PRODNUM;
85. select prodvolume into v_prodvolume from produit p
86. where p.PRODNUM = p_PRODNUM;
87. v_colvolume := v_qtecom*v_prodvolume;
88. v_colnum := SEQ_COLIS.nextval;
89. insert into colis VALUES (v_colnum, null, p_prodnun, un_commande.comnum, 'en cours de préparation', v_colv
    olume, v_qtecom);
90. SYS.DBMS_OUTPUT.PUT_LINE('Créer le colis ' || v_colnum || ' du produit ' || p_prodnun || ' pour la commande
    ' || un_commande.comnum);
91. RG_TERMINER_COMMANDE (un_commande.comnum, p_prodnun);
92. RG_DISTRIBUTION_COLIS(SEQ_COLIS.currval, v_colvolume);

```

```

93.         end loop;
94.     end if;
95. else
96.     SYS.DBMS_OUTPUT.PUT_LINE('Le produit a été déjà reçu');
97.     v_qterefus := p_QTELIV;
98. end if;
99. end if;
100. close c_qtecomglob;
101. -- chaque fois après nous avons reçu un livraison, nous changons la quantité refusée.
102. update livrer l set qterefus = v_qterefus where l.BONLIVNUM = p_BONLIVNUM and l.PRODNUM = p_PRODNUM;
103. SYS.DBMS_OUTPUT.PUT_LINE('Contrôle de quantité est fini');
104. END RG_PROD_QTE_LIV;

```

#### 2-2-1-5 : CREATION\_CHARGEMENT

Cette procédure est pour vérifier qu'est-ce que tous les produits dans le même secteur sont pris en charge.

```

1. create or replace PROCEDURE CREATION_CHARGEMENT
2. (
3.     P_CHARNUM IN NUMBER
4. )AS
5.     v_secnum NUMBER;
6.     v_colnum NUMBER;
7.     v_secnom VARCHAR2(32);
8.     cursor c_touslescolis is select col.colnum, m.secnum
9.         from colis col, commande com, magasin m
10.        where col.comnum = com.comnum
11.        and com.magnum = m.magnum
12.        and coletat = 'en cours de préparation';
13.     cursor c_colissec is select col.colnum
14.        from colis col, commande com, magasin m
15.        where col.comnum = com.comnum
16.        and com.magnum = m.magnum
17.        and m.secnum = v_secnum;
18. BEGIN
19.     open c_touslescolis;
20.     fetch c_touslescolis into v_colnum, v_secnum;
21.     if c_touslescolis%found then
22.         for un_colis in c_colissec loop
23.             update colis set charnum = P_CHARNUM, coletat = 'prise en charge' where colnum = un_colis.colnum;
24.         end loop;
25.         select secquartier into v_secnom from secteur where secnum = v_secnum;
26.         SYS.DBMS_OUTPUT.PUT_LINE('Tous les colis de ' || v_secnom || ' sont pris en charge');
27.     end if;
28.     close c_touslescolis;
29. END CREATION_CHARGEMENT;

```

### 2-2-1-6 : RG\_DISTRIBUTION\_COLIS

C'est le processus de traitement des colis, s'il existe de chargement avec le même secteur qui possède l'espace disponible, nous allons ajouter ce colis dans le chargement concerné.

```
1.  create or replace PROCEDURE RG_DISTRIBUTION_COLIS
2.  (
3.    P_COLNUM IN NUMBER
4.    , P_COLVOLUME IN NUMBER
5.  ) AS
6.    v_secteur_nouveau_colis NUMBER;
7.    v_charnum NUMBER;
8.    v_secteur_chargement NUMBER;
9.    v_colis_chargement NUMBER;
10.   v_colvolume_chargement NUMBER;
11.   v_somme_volume_chargement NUMBER;
12.   v_capacite_chargement NUMBER;
13.   v_charger BOOLEAN;
14.   -- au debut de chaque jour, on va mettre à jour des états des chargements
15.   -- chaque jour on crée des chargements manuellement, chaque chargement concerne qu'un seul secteur
16.   -- chaque jour un camion est affecté sur un seul chargement
17.   -
18.   tous les chargements du jour qui n'atteignent pas sa capacité sont en l'état de 'en cours', les autres sont en l'état de 'expédié'
19.   -- tous les chargements précédents passent en l'état de 'expédié' : on fait les livraisons par jour
20.   -- trouver tous les chargements
21.   cursor c_chargement is select charnum from chargement
22.   where CHARETAT = 'en cours';
23.   -- trouver des colis qui sont déjà dans le chargement
24.   cursor c_colis_chargement is select colnum, colvolume, secnum from colis col, commande com, magasin m
25.   where col.comnum = com.comnum and com.magnum = m.magnum and col.charnum = v_charnum;
26.   -- trouver des chargements vide
27.   cursor c_chargement_vide is select cha.charnum from chargement cha
28.   where cha.charetat = 'en cours'
29.   and cha.charnum not in (select co.charnum from colis co
30.   where co.charnum is not null);
31. BEGIN
32.   v_charger := false;
33.   v_somme_volume_chargement := 0;
34.   select secnum into v_secteur_nouveau_colis from colis col1, commande com1, magasin m1
35.   where col1.colnum = p_colnum and col1.comnum = com1.comnum and com1.magnum = m1.magnum;
36.   for un_chargement in c_chargement loop
37.     v_charnum := un_chargement.charnum;
38.     -- trouver la somme de volume des colis qui sont déjà dans ce chargement
39.     open c_colis_chargement;
40.     fetch c_colis_chargement into v_colis_chargement, v_colvolume_chargement, v_secteur_chargement;
41.     if v_secteur_chargement = v_secteur_nouveau_colis then
42.       v_charger := true;
43.       while c_colis_chargement%found loop
44.         v_somme_volume_chargement := v_somme_volume_chargement + v_colvolume_chargement;
45.         fetch c_colis_chargement into v_colis_chargement, v_colvolume_chargement, v_secteur_chargement;
46.       end loop;
47.       close c_colis_chargement;
48.       -- trouver la capacité de camion affecté sur ce chargement
49.       select modcapacite into v_capacite_chargement from chargement ch, camion cam, modele mo
50.       where ch.charnum = un_chargement.charnum and ch.camnum = cam.camnum and cam.modnum = mo.modnum;
51.       -- vérifier s'il y a d'espace suffisant pour ce colis
52.       if v_capacite_chargement - v_somme_volume_chargement >= p_colvolume then
53.         -- Cas 1 : exist le chargement avec le même secteur avec d'espace suffisant, on ajoute ce colis dans ce chargement
54.         update colis set charnum = un_chargement.charnum, coletat = 'prise en charge' where colnum = p_colnum;
55.         sys.dbms_output.put_line('colis ' || p_colnum || ' bien prise en charge');
56.         exit;
57.       else
58.         -- Cas 2 : exist le chargement avec le même secteur sans d'espace suffisant, il faut attendre
```

```

59.      sys.dbms_output.put_line(p_colnum || ' pas d'espace suffisant, il faut attendre');
60.      exit;
61.  end if;
62. end if;
63. close c_colis_chargement;
64. end loop;
65. if v_charger = false then
66.     -- pas de chargement avec le même secteur
67.     for un_chargement_vide in c_chargement_vide loop
68.         Cas 3 : n'existe pas de chargement pour ce secteur mais il existe de chargement vide(sans colis), on ajoute ce colis sur ce c
chargement vide
69.         update colis set charnum = un_chargement_vide.charnum, coletat = 'prise en charge' where colnum = p_colnum;
70.         sys.dbms_output.put_line('colis ' || p_colnum || ' bien prise en charge dans un chargement vide');
71.         v_charger := true;
72.         exit;
73.     end loop;
74. end if;
75. END RG_DISTRIBUTION_COLIS;

```

#### 2-2-1-7 : COLIS LIVRE

C'est pour modifier la quantité de stock de magasin, et vérifier l'état de commande et de chargement après un colis est livré.

```

1.  create or replace PROCEDURE COLIS_LIVRE
2.  (
3.      P_CHARNUM IN NUMBER,
4.      P_COLNUM IN NUMBER,
5.      P_COMNUM IN NUMBER,
6.      P_PRODNUM IN NUMBER,
7.      P_QTELIV IN NUMBER,
8.      P_DATELIV IN DATE
9.  ) AS
10.     v_prodlivre NUMBER;
11.     v_magnum NUMBER;
12.     v_colnum NUMBER;
13.     cursor c_prodlivre is select co.prodnum
14.         from concerner co
15.         where co.comnum = p_comnum
16.         and co.prodnum <> p_prodnum
17.         and co.prodnum not in (select colis.prodnum
18.             from colis
19.             where colis.comnum = p_comnum
20.             and colis.coletat = 'livré');
21.     cursor c_chargement is select colnum
22.         from colis
23.         where colnum <> p_colnum
24.         and charnum = p_charnum
25.         and coletat <> 'livré';
26. BEGIN
27.     --modifier qtestock
28.     select magnum into v_magnum from commande where comnum = p_comnum;
29.     update stocker set qtestock = qtestock + p_qteliv where magnum = v_magnum and prodnum = p_prodnum;
30.     --vérifier les colis reçus, si tous les colis de la commande sont bien reçus, modifier l'état de la commande
31.     open c_prodlivre;
32.     fetch c_prodlivre into v_prodlivre;
33.     if c_prodlivre%rowcount = 0 then
34.         update commande set cometat = 'livré', comdateliv = p_dateliv where comnum = p_comnum;
35.     end if;
36.     --vérifier l'état du chargement, si tous les colis du chargement sont bien livrés, modifier l'état du chargement
37.     open c_chargement;
38.     fetch c_chargement into v_colnum;
39.     if c_chargement%rowcount = 0 then
40.         update chargement set charetat = 'livré' where charnum = p_charnum;
41.     end if;

```



42. **END** COLIS\_LIVRE;

## 2-2-2 : Déclencheurs

### 2-2-2-1 : CREATION\_COMMANDE

Quand nous créons une commande, comnum est créé automatiquement en fonction de séquence et son état initial est 'en cours de constitution'.

```
1. create or replace TRIGGER CREATION_COMMANDE
2. BEFORE INSERT ON COMMANDE
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   :N.comnum := seq_commande.nextval;
7.   :N.cometat := 'en cours de constitution';
8. END;
```

### 2-2-2-2 : INSERER\_LIGNE\_COMMANDE

Avant nous allons ajouter une ligne de commande, il faut vérifier l'état de commande et l'espace de stock, s'il la commande n'est pas dans l'état de 'en cours de constitution' ou il n'y pas d'espace suffisant pour le stocker, nous ne pouvons pas ajouter ce produit.

```
1. create or replace TRIGGER INSERER_LIGNE_COMMANDE
2. BEFORE INSERT ON CONCERNER
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. DECLARE
6.   v_res_qte boolean;
7.   v_res_etat VARCHAR2(32);
8. BEGIN
9.   select cometat into v_res_etat from commande where comnum = :n.comnum;
10.  v_res_qte := rg_qte_commande(:n.comnum, :n.prodnum, :n.qtecom);
11.  if v_res_etat != 'en cours de constitution' then
12.    raise_application_error(-20101, 'Vous ne pouvez plus modifier cette commande');
13.  else
14.    if not v_res_qte then
15.      raise_application_error(-20101, 'La quantité commandée est hors limitée');
16.    end if;
17.  end if;
18. END;
```

### 2-2-2-3 : INSERER\_LIGNE\_COMGLOB

Avant nous ajoutons un produit dans la commande globale, nous exécutons la procédure RG\_PREPARATION\_COMMANDE pour vérifier l'existence des produits rests d'une commande qui sont en attente de factorisation, si non, nous allons changer l'état de la commande en 'en cours de préparation'.

```
1. create or replace TRIGGER INSERER_LIGNE_COMGLOB
2. BEFORE INSERT ON ASSOCIER
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   RG_PREPARATION_COMMANDE(:N.comnum, :N.prodnum);
7. END;
```

#### 2-2-2-4 : FACTORISATION\_COMMANDE

Quand l'état d'une commande passe en 'en cours', le système va exécuter la procédure RG\_FACTORISATION\_COMMANDE pour commencer de traiter tous les produits dans cette commande dans des commandes globales.

```
1. create or replace TRIGGER FACTORISATION_COMMANDE
2. INSTEAD OF UPDATE ON VIEW_COMMANDE
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   if :n.cometat = 'en cours' then
7.     update commande set cometat = 'en cours' where comnum = :n.comnum;
8.     RG_FACTORISATION_COMMANDE(:n.comnum);
9.   end if;
10. END;
```

#### 2-2-2-5 : BONLIVNUM\_AUTO

Quand le fournisseur fait des bons livraison, s'il n'a pas précisé le numéro, le numéro de bon livraison sera être créé automatiquement en fonction de séquence.

```
1. create or replace TRIGGER BONLIVNUM_AUTO
2. BEFORE INSERT ON BONLIVRAISON
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   if :n.bonlivnum is null then
7.     :n.bonlivnum := seq_bonliv.nextval;
8.   end if;
9. END;
```

#### 2-2-2-6 : CONTROLE\_QTE\_BONLIV

Après le fournisseur fait le bon livraison et quand nous avons reçu le produit concerné, nous allons remplir la table de fournir avec le bonlivnum, prodnum, et noter la quantité livré, après la contrôle, nous allons mettre à jour la quantité de refus.

```
1. create or replace TRIGGER CONTROLE_QTE_BONLIV
2. INSTEAD OF INSERT ON VIEW_LIVRER
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   insert into livrer values (:n.prodnum, :n.bonlivnum, :n.qteliv, 0);
7.   RG_PROD_QTE_LIV(:n.prodnum, :n.bonlivnum, :n.qteliv);
8. END;
```

#### 2-2-2-7 : CHARGEMENTNUM\_AUTO

Après avoir créé des chargements, nous allons chercher tous les commandes en attente de prise en charge (en l'état 'en cours de préparation'), et les classifier selon des différents secteurs et ensuite les ajouter dans des chargements.

```
1. create or replace TRIGGER CHARGEMENTNUM_AUTO
2. AFTER INSERT ON CHARGEMENT
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. BEGIN
6.   CREATION_CHARGEMENT(:n.charnum);
7. END;
```

#### 2-2-2-8 : LIVRAISON\_COLIS

Quand nous avons livré un colis, nous notons la date livrée pour mettre à jour l'attribut de comlivdate dans la table Commande et modifier la quantité de stock de magasin, et vérifier l'état de commande et de chargement. Si tous les colis dans cette commande sont livrés, l'état de cette commande passe en 'livrée'. Si tous les colis d'un chargement sont livrés, l'état de ce chargement passe en 'livré' en même temps.

```
1. create or replace TRIGGER LIVRAISON_COLIS
2. INSTEAD OF UPDATE ON VIEW_COLIS
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. DECLARE
6.   v_dateliv DATE;
7. BEGIN
8.   if :n.coletat = 'livré' then
9.     update colis set COLETAT=:n.coletat
10.    where colnum = :n.colnum;
11.    v_dateliv := sysdate;
12.    COLIS_LIVRE(:n.charnum, :n.colnum, :n.comnum, :n.prodnum, :n.qteliv, v_dateliv);
13.   else
14.     null;
15.   end if;
16. END;
```

#### 2-2-2-9 : EXPEDIER\_CHARGEMENT

Quand nous expédions un chargement, l'état de tous les colis de ce chargement passent en 'expédié' en même temps.

```
1. create or replace TRIGGER EXPEDIER_CHARGEMENT
2. AFTER UPDATE OF CHARETAT ON CHARGEMENT
3. REFERENCING OLD AS A NEW AS N
4. FOR EACH ROW
5. DECLARE
6.   cursor c_colis is select colnum from colis where charnum = :n.charnum;
7. BEGIN
8.   if :n.charetat = 'expédié' then
9.     for un_colis in c_colis loop
10.      update colis set coletat = 'expédié' where colnum = un_colis.colnum;
11.    end loop;
12.   end if;
13. END;
```

### 2-2-3 : Fonctions

Cette fonction est pour vérifier que la quantité commandée d'un produit ne dépasse pas l'espace disponible pour le stocker (l'espace disponible est la quantité max moins de la stock)

#### 2-2-3-1 : RG\_QTE\_COMMANDE

```
1. create or replace FUNCTION RG_QTE_COMMANDE
2. (P_COMNUM IN NUMBER
3. , P_PRODNUM IN NUMBER
4. , P_QTECOM IN NUMBER
5. ) RETURN boolean AS
6. res boolean;
7. p_magnum number;
8. p_qtestock number;
9. p_qtemax number;
10. BEGIN
11.   select qtestock, qtemax into p_qtestock, p_qtemax from stocker s, commande c
12.   where s.magnum = c.magnum
13.     and s.prodnum = p_prodnum
14.     and c.comnum = p_comnum;
15.   if p_qtecom <= p_qtemax - p_qtestock then
16.     return true;
17.   else
18.     return false;
19.   end if;
20. END RG_QTE_COMMANDE;
```

### 2-2-4 : Séquences

#### 2-2-4-1 : SEQ\_CHARGEMENT

```
1. CREATE SEQUENCE "CHEF_GUO"."SEQ_CHARGEMENT" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE ;
```

#### 2-2-4-2 : SEQ\_COMMANDE

```
1. CREATE SEQUENCE "CHEF_GUO"."SEQ_COMMANDE" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE ;
```

#### 2-2-4-3 : SEQ\_COMGLOB

```
1. CREATE SEQUENCE "CHEF_GUO"."SEQ_COMGLOB" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE ;
```

#### 2-2-4-4 : SEQ\_COLIS

```
1. CREATE SEQUENCE "CHEF_GUO"."SEQ_COLIS" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE ;
```

#### 2-2-4-5 : SEQ\_BONLIV

```
1. CREATE SEQUENCE "CHEF_GUO"."SEQ_BONLIV" MINVALUE 1 MAXVALUE 9999999999 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE ;
```

## Partie III : Programmation PL/SQL

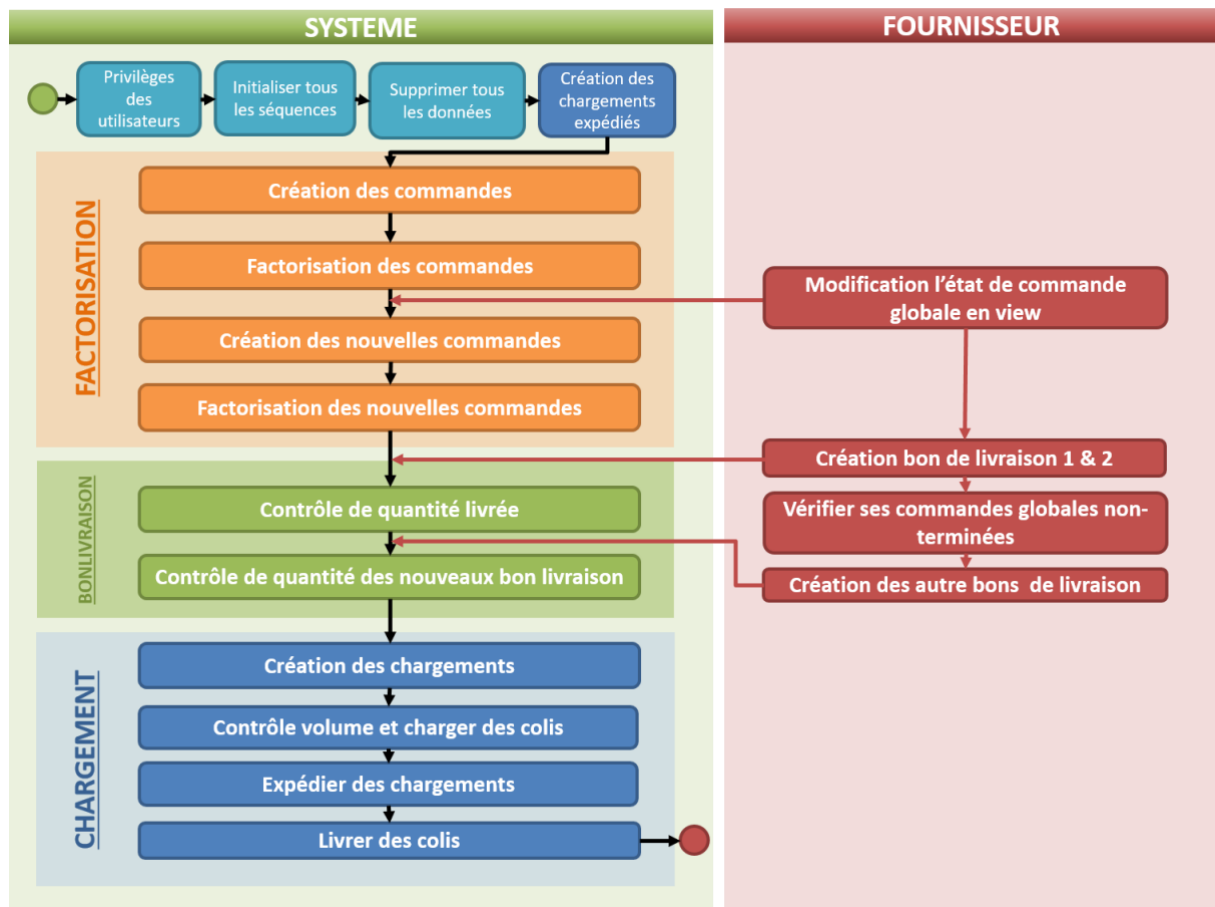
### 3-1 : Environnement de test

Après avoir créé 5 produits, 6 secteurs, 6 magasins, et 11 fournisseurs dans la base de données, nous avons inséré dans le table STOCKER pour chaque magasin des produit avec la quantité stocké, la quantité d'alerte et la quantité maximale. Nous avons lié les fournisseurs avec les produits, le prix et le seuil.

```
1. INSERT INTO PRODUIT VALUES ('1', 'jus d'orange', '1,5');
2. INSERT INTO PRODUIT VALUES ('2', 'yaourt', '1');
3. INSERT INTO PRODUIT VALUES ('3', 'chocolat noir', '0,2');
4. INSERT INTO PRODUIT VALUES ('4', 'pain', '0,5');
5. INSERT INTO PRODUIT VALUES ('5', 'cookies', '0,3');
6.
7. INSERT INTO FOURNISSEUR VALUES ('1', 'Andros');
8. INSERT INTO FOURNISSEUR VALUES ('2', 'Innocent');
9. INSERT INTO FOURNISSEUR VALUES ('3', 'Tropicana');
10. INSERT INTO FOURNISSEUR VALUES ('4', 'Nestlé');
11. INSERT INTO FOURNISSEUR VALUES ('5', 'Bonne Maman');
12. INSERT INTO FOURNISSEUR VALUES ('6', 'Danone');
13. INSERT INTO FOURNISSEUR VALUES ('7', 'Côté D'Or');
14. INSERT INTO FOURNISSEUR VALUES ('8', 'Milka');
15. INSERT INTO FOURNISSEUR VALUES ('9', 'Lindt');
16. INSERT INTO FOURNISSEUR VALUES ('10', 'Harry's');
17. INSERT INTO FOURNISSEUR VALUES ('11', 'LU');
18.
19. INSERT INTO SECTEUR VALUES ('1', 'Toulouse', 'Centre');
20. INSERT INTO SECTEUR VALUES ('2', 'Toulouse', 'Rive Gauche');
21. INSERT INTO SECTEUR VALUES ('3', 'Toulouse', 'Nord');
22. INSERT INTO SECTEUR VALUES ('4', 'Toulouse', 'Est');
23. INSERT INTO SECTEUR VALUES ('5', 'Toulouse', 'Sud-Est');
24. INSERT INTO SECTEUR VALUES ('6', 'Toulouse', 'Ouest');
25.
26. INSERT INTO MAGASIN VALUES ('1', '1', 'Carrefour');
27. INSERT INTO MAGASIN VALUES ('2', '2', 'Auchan');
28. INSERT INTO MAGASIN VALUES ('3', '3', 'Casino');
29. INSERT INTO MAGASIN VALUES ('4', '1', 'E.Leclerc');
30. INSERT INTO MAGASIN VALUES ('5', '2', 'Monoprix');
31. INSERT INTO MAGASIN VALUES ('6', '4', 'ChezJiayu');
32.
33. INSERT INTO STOCKER VALUES ('1', '1', '150', '80', '1000');
34. INSERT INTO STOCKER VALUES ('1', '2', '100', '50', '800');
35. INSERT INTO STOCKER VALUES ('1', '3', '90', '100', '800');
36. INSERT INTO STOCKER VALUES ('1', '4', '70', '30', '150');
37. INSERT INTO STOCKER VALUES ('1', '5', '80', '50', '900');
38. INSERT INTO STOCKER VALUES ('2', '1', '150', '90', '1000');
39. INSERT INTO STOCKER VALUES ('2', '2', '120', '80', '600');
40. INSERT INTO STOCKER VALUES ('2', '3', '150', '120', '700');
41. INSERT INTO STOCKER VALUES ('2', '4', '60', '40', '200');
42. INSERT INTO STOCKER VALUES ('2', '5', '130', '100', '800');
43. INSERT INTO STOCKER VALUES ('3', '1', '200', '80', '900');
44. INSERT INTO STOCKER VALUES ('3', '2', '200', '50', '850');
45. INSERT INTO STOCKER VALUES ('3', '3', '400', '100', '750');
46. INSERT INTO STOCKER VALUES ('3', '4', '80', '30', '200');
```

```
47. INSERT INTO STOCKER VALUES ('3','5','450','50','850');
48. INSERT INTO STOCKER VALUES ('4','1','250','100','850');
49. INSERT INTO STOCKER VALUES ('4','2','50','100','600');
50. INSERT INTO STOCKER VALUES ('4','3','150','300','900');
51. INSERT INTO STOCKER VALUES ('4','4','60','80','300');
52. INSERT INTO STOCKER VALUES ('4','5','150','120','700');
53. INSERT INTO STOCKER VALUES ('5','1','130','120','950');
54. INSERT INTO STOCKER VALUES ('5','2','200','60','700');
55. INSERT INTO STOCKER VALUES ('5','3','150','200','1000');
56. INSERT INTO STOCKER VALUES ('5','4','100','70','300');
57. INSERT INTO STOCKER VALUES ('5','5','300','150','700');
58. INSERT INTO STOCKER VALUES ('6','1','500','150','1000');
59. INSERT INTO STOCKER VALUES ('6','2','90','80','500');
60. INSERT INTO STOCKER VALUES ('6','3','150','200','1200');
61. INSERT INTO STOCKER VALUES ('6','4','50','100','900');
62. INSERT INTO STOCKER VALUES ('6','5','50','150','800');
63.
64. INSERT INTO FOURNIR VALUES (1,1,sysdate,1.6,400);
65. INSERT INTO FOURNIR VALUES (2,1,sysdate,1.5,600);
66. INSERT INTO FOURNIR VALUES (3,1,sysdate,1.7,500);
67. INSERT INTO FOURNIR VALUES (4,1,sysdate,1.2,800);
68. INSERT INTO FOURNIR VALUES (4,2,sysdate,0.61,500);
69. INSERT INTO FOURNIR VALUES (5,2,sysdate,0.66,200);
70. INSERT INTO FOURNIR VALUES (6,2,sysdate,0.79,300);
71. INSERT INTO FOURNIR VALUES (4,3,sysdate,0.91,200);
72. INSERT INTO FOURNIR VALUES (7,3,sysdate,0.88,500);
73. INSERT INTO FOURNIR VALUES (8,3,sysdate,0.81,700);
74. INSERT INTO FOURNIR VALUES (9,3,sysdate,0.99,400);
75. INSERT INTO FOURNIR VALUES (5,4,sysdate,0.57,150);
76. INSERT INTO FOURNIR VALUES (10,4,sysdate,0.55,120);
77. INSERT INTO FOURNIR VALUES (11,4,sysdate,0.8,100);
78.
79. INSERT INTO MODELE VALUES (1,'Mercedes-Benz Antos',2000)
80. INSERT INTO MODELE VALUES (2,'Mercedes-Benz Canter',2000)
81. INSERT INTO MODELE VALUES (3,'Alfa Romeo 50',2000)
82.
83. INSERT INTO CAMION VALUES (4,3,150000);
84. INSERT INTO CAMION VALUES (1,1,50000);
85. INSERT INTO CAMION VALUES (2,2,120000);
86. INSERT INTO CAMION VALUES (3,3,86230);
```

La figure au-dessous indique notre processus de test contenant tous les cas concernant tous les règles.



Nous avons divisé le processus de test en trois parties : factorisation, bonlivraison et chargement.

Avant commencer le test, nous initialisons tous les séquence et les données, qui nous permet de mieux préparer l'environnement de test. Nous créons des chargements expédiés, pour vérifier que le colis ne peut qu'être mis dans le chargement 'en cours'.

D'abord, le système crée des commandes et les factorise. Le fournisseur peut consulter ses commandes globales et modifier l'état de ces commandes en 'en cours de préparation'. Pour vérifier que le système ne peut pas répartir les produits dans les commande globale en état de 'en cours de préparation', le système crée des nouvelles commandes et les factorise après que le fournisseur change l'état de commande globale existantes en 'en cours de préparation'.

Ensuite le fournisseur crée des bons de livraison 1 & 2 concernant sa commande globale non terminé. Quand on reçoit le bon de livraison, le système passe le processus de contrôle de bon de livraison. Le fournisseur peut consulter tous ses bons de livraison avec la quantité livrée et la quantité refusée et aussi les produits en attente de livraison. Il crée des nouveaux bons de livraison, et le système les contrôle.

A la fin, le système crée des chargements, il prend des colis en attente automatiquement après le contrôle de volume de colis. Et puis le système modifie l'état des chargements en 'expédié'. Quand les colis sont livrés le système modifie l'état de colis.

## 3-2 : Résultat de test

Dans cette partie, nous démontrons notre code et le résultat de test.

### 3-2-1 Préparation de test

Avant commencer le test, nous initialisons les séquences et supprimons des données dynamiques.

```
1.  --initialiser tous les seq
2.  DROP sequence SEQ_BONLIV;
3.  CREATE SEQUENCE SEQ_BONLIV INCREMENT BY 1 START WITH 1 MAXVALUE 99999999999 MINVALUE 1;
4.  DROP sequence SEQ_CHARGEMENT;
5.  CREATE SEQUENCE SEQ_CHARGEMENT INCREMENT BY 1 START WITH 1 MAXVALUE 99999999999 MINVALUE 1;
6.  DROP sequence SEQ_COLIS;
7.  CREATE SEQUENCE SEQ_COLIS INCREMENT BY 1 START WITH 1 MAXVALUE 99999999999 MINVALUE 1;
8.  DROP sequence SEQ_COMGLOB;
9.  CREATE SEQUENCE SEQ_COMGLOB INCREMENT BY 1 START WITH 1 MAXVALUE 99999999999 MINVALUE 1;
10. DROP sequence SEQ_COMMANDE;
11. CREATE SEQUENCE SEQ_COMMANDE INCREMENT BY 1 START WITH 1 MAXVALUE 99999999999 MINVALUE 1;
12.
13. --supprimer tous les données
14. delete colis;
15. delete livrer;
16. delete bonlivraison;
17. delete concerner;
18. delete concernerglob;
19. delete associer;
20. delete COMMANDEGLOBALE;
21. delete commande;
22. delete chargement;
23. update fournisseur set histodate = sysdate where histodate = (select max(histodate) from fournisseur);
24. update stocker set qtestock =150 where magnum = 2 and prodnum = 1;
25. update stocker set qtestock =120 where magnum = 2 and prodnum = 2;
26. update stocker set qtestock =150 where magnum = 2 and prodnum = 3;
27. commit;
```

Le système crée 4 chargements expédié, pour le test de chargement de colis en cas que tous les chargements ont été déjà expédiés.

```
1.  --création des chargement expédié
2.  insert into chargement values(1,1,sysdate,'expédié');
3.  insert into chargement values(2,2,sysdate,'expédié');
4.  insert into chargement values(3,3,sysdate,'expédié');
5.  insert into chargement values(4,4,sysdate,'expédié');
6.  commit;
```



### 3-2-2 Factorisation

D'abord, le système crée 3 commandes. On considère que le numéro de commande est donné par le séquence automatiquement quand on crée une nouvelle commande et l'état de cette commande est 'en cours de constitution'. Le système insère chaque ligne de commande dans le table concerner, dès que tous les ligne sont bien insérées, il modifie l'état de commande en 'en cours'.

```

7. --création de commande
8. insert into commande values(null,2,sysdate,null,'en cours de constitution');
9. insert into concerner values(1,1,800);
10. insert into concerner values(1,2,300);
11. insert into concerner values(1,3,50);
12.
13. insert into commande values(null,3,sysdate,null,'en cours de constitution');
14. insert into concerner values(2,1,100);
15. insert into concerner values(2,2,250);
16.
17. insert into commande values(null,4,sysdate,null,'en cours de constitution');
18. insert into concerner values(3,3,170);
19. commit;
20. --résultat de création de commande
21. select com.comnum, co.prodnum, co.qtecom,com.magnum, com.comdate,com.comdateliv,com.cometat
22. from commande com, concerner co
23. where com.comnum = co.comnum;

```

Le tableau montre le résultat de création des commandes.

	...	PRODNUM	QTECOM	MAGNUM	COMDATE	COMDATELIV	COMETAT
1	1	1	800	2	13-JAN-19	(null)	en cours de constitution
2	1	2	300	2	13-JAN-19	(null)	en cours de constitution
3	1	3	50	2	13-JAN-19	(null)	en cours de constitution
4	2	1	100	3	13-JAN-19	(null)	en cours de constitution
5	2	2	250	3	13-JAN-19	(null)	en cours de constitution
6	3	3	170	4	13-JAN-19	(null)	en cours de constitution

Ensuite, le système modifie l'état de ces commande en 'en cours', ces commandes vont passer au processus de factorisation.

```

1. --factorisation de commande
2. update view_commande set cometat = 'en cours' where COMNUM = 1;
3. update view_commande set cometat = 'en cours' where COMNUM = 2;
4. update view_commande set cometat = 'en cours' where COMNUM = 3;
5. commit;

```

Le script d'output :

```
Commande 1
Produit1
Le fournisseur 4 est le moins cher
On atteint le seuil
Commande n'existe pas, créer une commande globale
Insérer une ligne de commande globale du produit 1
Insérer tous les commande concerné du produit 1 dans le table associer
Produit2
Le fournisseur 4 est le moins cher
On n'atteint pas le seuil
Produit3
Le fournisseur 8 est le moins cher
On n'atteint pas le seuil
Factorisation de la commande 1 est finie

Commande 2
Produit1
produit 1 --existe une commande globale, augmenter la qtecomglob
Produit2
Le fournisseur 4 est le moins cher
On atteint le seuil
Insérer une ligne de commande globale du produit 2
Tous les produit de la commande 2 sont factorisé. L'état de commande passe en 'en cours de préparation'.
Insérer tous les commande concerné du produit 2 dans le table associer
Factorisation de la commande 2 est finie

Commande 3
Produit3
Le fournisseur 8 est le moins cher
On n'atteint pas le seuil
stock alerte du produit 3
Le fournisseur 4 est le moins cher avec bon seuil du produit 3
Insérer une ligne de commande globale du produit 3
Tous les produit de la commande 3 sont factorisé. L'état de commande passe en 'en cours de préparation'.
Tous les produit de la commande 1 sont factorisé. L'état de commande passe en 'en cours de préparation'.
Insérer tous les commande concerné du produit 3 dans le table associer
Factorisation de la commande 3 est finie
```

Le résultat de factorisation:

1. **select** comglobnum,comnum, prodnum
2. **from** associer
3. **where** comnum in (1,2,3);

	COMGLOBNUM	COMNUM	PRODNUM
1	1	1	1
2	1	1	2
3	1	1	3
4	1	2	1
5	1	2	2
6	1	3	3

1. **select** comglobnum, prodnum, qtecomglob
2. **from** concernerglob
3. **where** comglobnum = 1;

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	1	1	900
2	1	2	550
3	1	3	220

Le fournisseur 4 qui est concerné à la commande globale 1 peut consulter trouver maintenant la commande globale 1 dans **view\_commandeglobale** et les produits en attente de livraison dans **view\_produit**.

	COMGLOBNUM	FOURNUM	COMGLOBDATE	COMGLOBETAT
1	1	4	13-JAN-19	en cours de constitution

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	1	1	900
2	1	2	550
3	1	3	220

Quand il commence à préparer les produits concernant cette commande, il modifie l'état de commande globale 1 en 'en cours de préparation' directement dans **view\_commandeglobale**, les données dans cette base de données vont être modifiée automatiquement grâce au procédure **ETAT\_PREPARATION** qui permet une transaction autonome.

	COMGLOBNUM	FOURNUM	COMGLOBDATE	COMGLOBETAT
1	1	4	13-JAN-19	en cours de préparation

Pour vérifier qu'on ne peut plus factoriser une commande dans les commande globale 'en cours de préparation', le système crée 2 nouvelles commandes et les factorise.

1. --création des nouvelles commandes
2. **insert into** commande **values**(4,5,sysdate,null,'en cours de constitution');
3. **insert into** commande **values**(5,6,sysdate,null,'en cours de constitution');
- 4.
5. **insert into** concerner **values**(4,1,800);
6. **insert into** concerner **values**(5,3,800);
7. **insert into** concerner **values**(5,4,200);
- 8.
9. --factorisation des nouvelles commandes
10. **update** view\_commande **set** cometat = 'en cours' **where** COMNUM = 4;

```
11. update view_commande set cometat = 'en cours' where COMNUM = 5;
12. commit;
```

Le résultat du script d'output :

Commande 4

Produit1

Le fournisseur 4 est le moins cher

On atteint le seuil

Commande n'existe pas, créer une commande globale

Insérer une ligne de commande globale du produit 1

Tous les produit de la commande 4 sont factorisé. L'état de commande passe en 'en cours de préparation'.

Insérer tous les commande concerné du produit 1 dans le table associer

Factorisation de la commande 4 est finie

Commande 5

Produit3

Le fournisseur 8 est le moins cher

On atteint le seuil

Commande n'existe pas, créer une commande globale

Insérer une ligne de commande globale du produit 3

Insérer tous les commande concerné du produit 3 dans le table associer

Produit4

Le fournisseur 10 est le moins cher

On atteint le seuil

Commande n'existe pas, créer une commande globale

Insérer une ligne de commande globale du produit 4

Tous les produit de la commande 5 sont factorisé. L'état de commande passe en 'en cours de préparation'.

Insérer tous les commande concerné du produit 4 dans le table associer

Factorisation de la commande 5 est finie

Le résultat de factorisation:

```
1. select comglobnum, comnum, prodnum
2. from associer
3. where comglobnum <> 1;
```

	COMGLOBNUM	COMNUM	PRODNUM
1	2	4	1
2	3	5	3
3	4	5	4

```
1. select comglobnum, prodnum, qtecomglob
2. from concernerglob
3. where comglobnum <> 1;
```

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	2	1	800
2	3	3	800
3	4	4	200

Les trois nouvelles commandes sont réparties en trois commandes globale avec des différents fournisseurs, parmi eux, le fournisseur de la commande globale 2 est le même que la commande globale 1. Donc le fournisseur 4 ne peut trouver que la commande globale 1 et 2 dans **view\_commandeglobale**, et les produits concernant (1,2,3) dans **view\_produit**.

	COMGLOBNUM	FOURNUM	COMGLOB...	COMGLOBETAT
1	1		4 13-JAN-19	en cours de préparation
2	2		4 13-JAN-19	en cours de constitution

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	1	1	900
2	1	2	550
3	1	3	220
4	2	1	800

### 3-2-3 Bonlivraison

Le fournisseur crée 2 bons de livraison.

```

1. --création bonlivraison 1, 2
2. insert into CHEF_GUO.bonlivraison values(null,1,sysdate);
3. insert into CHEF_GUO.bonlivraison values(null,1,sysdate);
4. commit;
```

Dans **view\_bonlivraison** le fournisseur trouve les colis qu'il a déjà émis.

	BONLIVNUM	PRODNUM	QTELIV	QTEREFUS
1	1	(null)	(null)	(null)
2	2	(null)	(null)	(null)

Dès la réception de bon de livraison, le système insère la quantité reçue (qteliv) dans le table livrer en fonction du produit et bonlivraison. La quantité refuse(qterefus) est mis à jour automatiquement en fonction de règle de contrôle de bon de livraison.

```

1. --contrôle de la quantité livrée
2. --recevoir le bonlivraison 1
3. --cas 1 : produit ne corresponde pas, refus = 500
4. insert into VIEW_LIVRER values (4,1,500);
5. --cas 2 : quantité insuffisante, refus = 500
6. insert into VIEW_LIVRER values (1,1,500);
7. --cas 5 : bonne auqntité
8. insert into VIEW_LIVRER values (2,1,550);
9. commit;
10.
11. --recevoir le bonlivraison 2
12. insert into VIEW_LIVRER values (3,2,220);
13. commit;
```

Le script d'output

```

Bonlivraison 1 : produit 4 quantité : 500
produit ne correspond pas
Controle de quantité est fini
```

Bonlivraison 1 : produit 1 quantité : 500  
quantité insuffisant  
Controle de quantité est fini  
Bonlivraison 1 : produit 2 quantité : 550  
quantité dépassant ou bon  
Créer le colis 1 du produit 2 pour la commande 1  
Créer le colis 2 du produit 2 pour la commande 2  
Controle de quantité est fini

Bonlivraison 2 : produit 3 quantité : 220  
quantité dépassant ou bon  
Créer le colis 3 du produit 3 pour la commande 1  
Créer le colis 4 du produit 3 pour la commande 3  
Tous les produit de la commande 3 sont placés dans colis  
Controle de quantité est fini

Lors de la réception de bon produit de la bonne quantité, le système crée des colis en fonction de produit et commande concernée.

	COLNUM	CHARNUM	PRODNUM	COMNUM	COLETAT	COLVOLUME	QTELIV
1	4	(null)	3	3	en cours de préparation	34	170
2	3	(null)	3	1	en cours de préparation	10	50
3	2	(null)	2	2	en cours de préparation	250	250
4	1	(null)	2	1	en cours de préparation	300	300

Les numéros de chargement (charnum) sont null, puisque à ce moment-là il n'existe pas de chargement disponible, tous ces colis doivent attendre de prise en chargement.

Le fournisseur trouve le mis à jour dans **view\_bonlivraison** avec la quantité refusée des bons de livraison reçus par la centrale. Dans **view\_produit**, il trouve les produits en attente de livrer.

	BONLIVNUM	PRODNUM	QTELIV	QTEREFUS
1	1	4	500	500
2	1	1	500	500
3	1	2	550	0
4	2	3	220	0

Il reste que 2 produit à livrer pour le fournisseur 4.

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	2	1	800
2	1	1	900

Donc le fournisseur crée deux nouveaux bons de livraison, mais quand le système les contrôle, on trouve qu'un produit de la même commande globale a été déjà bien reçu.

Le script d'output

Bonlivraison 3 : produit 1 quantité : 1000  
quantité dépassant ou bon  
la commande globale 1 est terminée  
Créer le colis 5 du produit 1 pour la commande 1  
Tous les produit de la commande 1 sont placés dans colis  
Créer le colis 6 du produit 1 pour la commande 2  
Tous les produit de la commande 1 sont placés dans colis  
Controle de quantité est fini

Bonlivraison 4 : produit 1 quantité : 900

Le produit a été déjà reçu

Controle de quantité est fini

Le résultat de contrôle

Le système crée des nouveaux colis et les données sont mis à jour pour le fournisseur.

	CO...	CHARNUM	PRODNUM	COMNUM	COLETAT	COLVOLUME	QTELIV
1	1	(null)	2	1	en cours de préparation	300	300
2	2	(null)	2	2	en cours de préparation	250	250
3	3	(null)	3	1	en cours de préparation	10	50
4	4	(null)	3	3	en cours de préparation	34	170
5	5	(null)	1	1	en cours de préparation	1200	800
6	6	(null)	1	2	en cours de préparation	150	100

	BOHLIVNUM	PRODNUM	QTELIV	QTEREFUS
1	3	1	1000	100
2	4	1	900	900
3	1	4	500	500
4	1	1	500	500
5	1	2	550	0
6	2	3	220	0

Il reste la commande globale 2 à livrer pour le fournisseur 4.

	COMGLOBNUM	PRODNUM	QTECOMGLOB
1	2	1	800

L'état de la commande globale 1 a passé en 'terminée', puisque tous les produit de cette commande ont été déjà livré.

	COMG...	FOURNUM	COMGLOB...	COMGLOBETAT
1	2	4	13-JAN-19	en cours de constitution
2	1	4	13-JAN-19	terminée

Les commande concernant commande globale 1 sont terminées, puisque tous les produits de ces commandes ont été placés dans des colis.

1. **select** a.comglobnum, co.comnum, c.cometat, co.prodnum, col.colnum
2. **from** concerner co, colis col, associer a, commande c
3. **where** co.comnum = col.comnum(+)
4. **and** co.prodnum = col.prodnum(+)
5. **and** co.comnum = a.comnum(+)
6. **and** co.prodnum = a.prodnum(+)
7. **and** c.comnum = co.comnum
8. **order by** comnum, prodnum;

	COMGLOBNUM	COMNUM	COMETAT	PRODNUM	COLNUM
1	1	1	terminée	1	5
2	1	1	terminée	2	1
3	1	1	terminée	3	3
4	1	2	terminée	1	6
5	1	2	terminée	2	2
6	1	3	terminée	3	4
7	2	4	en cours de préparation	1	(null)
8	3	5	en cours de préparation	3	(null)
9	4	5	en cours de préparation	4	(null)

### 3-2-4 Chargement et Livraison

Le système crée 4 chargements, les colis en attente de charger, dans ce cas-là qui les colis concernant la commande globale 1, vont être mis dans ces chargement automatiquement en fonction de leur secteur.

```

1. --création des chargements et prendre les colis non chargés
2. insert into chargement values(5,1,sysdate,'en cours');
3. insert into chargement values(6,2,sysdate,'en cours');
4. insert into chargement values(7,3,sysdate,'en cours');
5. insert into chargement values(8,4,sysdate,'en cours');
6. commit;
7.
8. select m.secnum, col.charnum, col.colnum, c.comnum
9. from colis col, commande c, magasin m
10. where col.comnum = c.comnum
11. and c.magnum = m.magnum;
```

	SECNUM	CHARNUM	COLNUM	COMNUM
1	2	5	5	1
2	2	5	1	1
3	2	5	3	1
4	3	6	6	2
5	3	6	2	2
6	1	7	4	3

Et puis, la centrale reçoit des nouveaux bons de livraison par rapport à commande globale 2, 3 et 4. Parce que tous ces bons de livraison sont bon avec la quantité, le système crée des colis automatiquement. Ces colis vont être mis dans les chargements 5, 6, 7 et 8, s'il y a de l'espace disponible.

```

1. --contrôle de quantité et charger les colis automatiquement
2. insert into bonlivraison values(null,2,sysdate);
3. insert into VIEW_LIVRER values (1,5,800);
4.
5. insert into bonlivraison values(null,3,sysdate);
6. insert into VIEW_LIVRER values (3,6,800);
7.
8. insert into bonlivraison values(null,4,sysdate);
9. insert into VIEW_LIVRER values (4,7,200);
```

Le script d'output



Bonlivraison 5 : produit 1 quantité : 800  
quantité dépassant ou bon  
la commande globale 2 est terminée  
Créer le colis 7 du produit 1 pour la commande 4  
Tous les produit de la commande 1 sont placés dans colis  
7 pas d'espace suffisant, il faut attendre  
Controle de quantité est fini

Bonlivraison 6 : produit 3 quantité : 800  
quantité dépassant ou bon  
la commande globale 3 est terminée  
Créer le colis 8 du produit 3 pour la commande 5  
colis 8 bien prise en charge dans un chargement vide  
Controle de quantité est fini

Bonlivraison 7 : produit 4 quantité : 200  
quantité dépassant ou bon  
la commande globale 4 est terminée  
Créer le colis 9 du produit 4 pour la commande 5  
Tous les produit de la commande 4 sont placés dans colis  
colis 9 bien prise en charge  
Controle de quantité est fini








1. `select m.secnum, col.charnum, col.colnum, c.comnum`
2. `from colis col, commande c, magasin m`
3. `where col.comnum = c.comnum`
4. `and c.magnum = m.magnum;`

	...	CHARNUM	COLNUM	COMNUM
1	2	5	5	1
2	2	5	1	1
3	2	5	3	1
4	3	6	6	2
5	3	6	2	2
6	1	7	4	3
7	2	(null)	7	4
8	4	8	8	5
9	4	8	9	5

Le colis 7 n'est pas mis dans le chargement, puisqu'il y a pas d'espace disponible dans le chargement qui s'adresse au secteur 2.

Lorsque le chargement 6 passe en état 'expédié', tous les colis concernant sont modifiés en 'expédié'

1. `--expédier le chargement`
2. `update chargement set CHARETAT = 'expédié' where charnum = 5;`








	 2 COLNUM	 CHARNUM	 PRODNUM	 1 C. .	 COLETAT	 COLVOLUME	 QTELIV
1	1	5	2	1	expédié	300	300
2	3	5	3	1	expédié	10	50
3	5	5	1	1	expédié	1200	800
4	2	6	2	2	prise en charge	250	250
5	6	6	1	2	prise en charge	150	100
6	4	7	3	3	prise en charge	34	170
7	7	(null)	1	4	en cours de préparation	1200	800
8	8	8	3	5	prise en charge	160	800
9	9	8	4	5	prise en charge	100	200

A la fin, le système modifie l'état des colis dans chargement 5 en 'livré', quand ils sont livrés. En même temps, le système vérifie si tous les colis concernant la commande et le chargement sont bien livrés. Dès que le dernier colis d'une commande est livré, le système modifie l'état de cette commande et met à jour la date de livraison de cette commande. Dès que le dernier colis d'un chargement est livré, le système modifie l'état de ce chargement en 'livré'.






```

1. --livrer des colis
2. update view_colis set coletat = 'livré' where colnum in (select co.colnum
3.                                     from colis co
4.                                     where co.charnum = 5);





```

	 2 COLNUM	 CHARNUM	 PRODNUM	 1 COMNUM	 COLETAT	 COLVOLUME	 QTELIV
1	1	5	2	1	livré	300	300
2	3	5	3	1	livré	10	50
3	5	5	1	1	livré	1200	800
4	2	6	2	2	prise en charge	250	250
5	6	6	1	2	prise en charge	150	100
6	4	7	3	3	prise en charge	34	170
7	7	(null)	1	4	en cours de préparation	1200	800
8	8	8	3	5	prise en charge	160	800
9	9	8	4	5	prise en charge	100	200

Lors de la livraison du dernier colis (5) du chargement 1 est livré, l'état de la commande 1 passe en 'livré', la date de livraison de commande 1 est la date de livraison du colis 5.

	 COMNUM	 MAGNUM	 COMDATE	 COMDATELIV	 COMETAT
1	5	6	13-JAN-19	(null)	terminée
2	1	2	13-JAN-19	13-JAN-19	livré
3	2	3	13-JAN-19	(null)	terminée
4	3	4	13-JAN-19	(null)	terminée
5	4	5	13-JAN-19	(null)	terminée

En même temps, tous les colis de chargement 5 sont livrés, donc le chargement 5 passe en état 'livré'.

	 1 CHARNUM	 CAMNUM	 CHARDATE	 CHARETAT
1	1	1	13-JAN-19	expédié
2	2	2	13-JAN-19	expédié
3	3	3	13-JAN-19	expédié
4	4	4	13-JAN-19	expédié
5	5	1	13-JAN-19	livré
6	6	1	13-JAN-19	en cours
7	7	1	13-JAN-19	en cours
8	8	1	13-JAN-19	en cours

Jusqu'à là, tous les règles sont bien passés le test.

## CONCLUSION

Le projet nous a permis de réfléchir sur de nombreux aspects différents. Nous avons mieux compris la manipulation de privilège aux utilisateurs des différents types, et mieux su des méthodes alternatives et répétitives pour traiter des différentes conditions de processus. Nous avons dû faire des choix entre les points importants et les détails. Nous avons fixé quelques limites pour améliorer les processus de cette application. Nous pensons être arrivé à un bon compromis entre réalisme et créativité.