



Ressourcesinformatiques

# Oracle 12c

## SQL, PL/SQL, SQL\*Plus

Jérôme GABILLAUD

Téléchargement  
[www.editions-eni.fr](http://www.editions-eni.fr)



Les éléments à télécharger sont disponibles à l'adresse suivante :  
**<http://www.editions-eni.fr>**  
Saisissez la référence ENI de l'ouvrage **RI12CORA** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

## Avant-propos

### Chapitre 1

#### Modèle relationnel

1. La gestion des données . . . . .	15
1.1 Généralités sur les fichiers . . . . .	15
1.2 Organisations classiques de fichiers . . . . .	16
2. Le modèle relationnel . . . . .	18
2.1 Concepts et définitions . . . . .	18
2.1.1 Domaine . . . . .	19
2.1.2 Produit cartésien . . . . .	19
2.1.3 Relation . . . . .	20
2.2 Principales règles . . . . .	21
3. L'algèbre relationnelle . . . . .	23
3.1 Opérateurs . . . . .	24
3.1.1 Union . . . . .	24
3.1.2 Intersection . . . . .	25
3.1.3 Différence . . . . .	25
3.1.4 Restriction . . . . .	25
3.1.5 Projection . . . . .	26
3.1.6 Produit cartésien . . . . .	28
3.1.7 Jointures . . . . .	28
3.1.8 Calculs élémentaires . . . . .	30
3.1.9 Calculs d'agrégats . . . . .	30

3.2	Étapes de résolution d'un problème . . . . .	32
3.2.1	Analyser le besoin . . . . .	32
3.2.2	Établir la vue . . . . .	32
3.2.3	Ordonnancer et exprimer les opérations . . . . .	32

## Chapitre 2

### SQL

1.	Généralités . . . . .	37
1.1	Composants de la base logique : les objets SQL . . . . .	39
1.1.1	La gestion des données . . . . .	39
1.1.2	Le stockage physique . . . . .	40
1.1.3	Le stockage d'instructions . . . . .	40
1.1.4	La gestion des utilisateurs . . . . .	40
1.1.5	Dénomination des objets . . . . .	41
1.2	Catégories d'instructions . . . . .	41
1.2.1	DDL (Data Definition Language) . . . . .	42
1.2.2	DML (Data Manipulation Language) . . . . .	42
1.2.3	TCL (Transaction Control language) . . . . .	42
1.2.4	SCL (Session Control language) . . . . .	42
1.2.5	Embedded SQL . . . . .	43
2.	Description des objets . . . . .	43
2.1	Les types de données . . . . .	43
2.2	Création d'une table . . . . .	46
2.2.1	Contraintes de colonne . . . . .	47
2.2.2	Contraintes de table (portant sur plusieurs colonnes) . . . . .	48
2.2.3	Complément sur les contraintes . . . . .	49
2.2.4	Dénomination des contraintes . . . . .	49
2.2.5	Colonne virtuelle . . . . .	52
2.2.6	Colonne identité . . . . .	53
2.2.7	Colonne invisible . . . . .	57
2.3	Suppression d'une table . . . . .	58

- 2.4 Modification d'une table ..... 60
  - 2.4.1 Ajout ou modification de colonnes ..... 60
  - 2.4.2 Ajout d'une contrainte de table ..... 62
  - 2.4.3 Suppression d'une contrainte ..... 64
  - 2.4.4 Activation/désactivation d'une contrainte ..... 64
  - 2.4.5 Modification d'une contrainte ..... 66
  - 2.4.6 Suppression de colonnes ..... 67
  - 2.4.7 Changement de nom d'une table ..... 69
  - 2.4.8 Mettre une table en lecture seule  
ou en lecture-écriture ..... 69
- 2.5 Restauration d'une table ..... 70
- 2.6 Gestion des index ..... 73
  - 2.6.1 Création d'un index ..... 73
  - 2.6.2 Suppression d'un index ..... 74
- 3. Manipulation des données ..... 74
  - 3.1 Les instructions ..... 74
    - 3.1.1 Expressions ..... 74
    - 3.1.2 Opérateurs ..... 75
    - 3.1.3 Conditions ..... 75
    - 3.1.4 Fonctions ..... 76
  - 3.2 Création de lignes ..... 89
  - 3.3 Suppression de lignes ..... 92
  - 3.4 Modification de lignes ..... 94
  - 3.5 Extraction des données ..... 95
  - 3.6 Contrôle des transactions ..... 96
    - 3.6.1 Validation de la transaction ..... 96
    - 3.6.2 Annulation des modifications ..... 97
    - 3.6.3 Déclaration d'un point de contrôle ..... 97
    - 3.6.4 Accès simultané aux données ..... 99
    - 3.6.5 Vérification des contraintes en fin de transaction ... 102

4.	Traduction de l'algèbre relationnelle . . . . .	104
4.1	Opérations . . . . .	104
4.1.1	Restriction. . . . .	104
4.1.2	Calculs élémentaires. . . . .	105
4.1.3	Projection . . . . .	106
4.1.4	Calculs d'agrégats . . . . .	108
4.1.5	Fonctions de groupe . . . . .	108
4.1.6	Fonctions analytiques . . . . .	112
4.1.7	Restrictions sur agrégat . . . . .	116
4.1.8	Produit cartésien. . . . .	116
4.1.9	Jointures . . . . .	118
4.1.10	Jointures externes. . . . .	119
4.1.11	Union, intersection, différence . . . . .	121
4.2	Traitement du résultat. . . . .	122
4.2.1	Le tri . . . . .	122
4.2.2	La sauvegarde . . . . .	124
4.2.3	Énumérer toutes les possibilités d'un calcul d'agrégats . . . . .	128
4.3	L'instruction MERGE. . . . .	131
5.	SQL avancé . . . . .	133
5.1	Les objets. . . . .	134
5.1.1	Les objets View (vue). . . . .	134
5.1.2	Les objets Schema. . . . .	140
5.1.3	Les objets Synonym . . . . .	144
5.1.4	Les objets Sequence . . . . .	146
5.2	Les requêtes complexes . . . . .	149
5.2.1	Éléments de syntaxe. . . . .	149
5.2.2	Les sous-requêtes . . . . .	152
5.2.3	Les requêtes hiérarchiques . . . . .	155
5.2.4	Pivoter les données. . . . .	158
5.2.5	Limiter le résultat d'une requête . . . . .	163
5.3	Verrouillage des tables . . . . .	167
5.4	Les commentaires . . . . .	170

- 5.5 Informations sur les objets du schéma . . . . . 171
- 5.6 Fonctionnalités spécifiques . . . . . 173
- 5.7 Les expressions régulières . . . . . 176

**Chapitre 3**  
**SQL\*Plus et SQL Developer**

- 1. Utilisation de SQL\*Plus . . . . . 185
  - 1.1 Connexion et déconnexion . . . . . 186
    - 1.1.1 Lancement du programme . . . . . 186
    - 1.1.2 Connexion après lancement . . . . . 188
    - 1.1.3 Changement du mot de passe . . . . . 188
    - 1.1.4 Déconnexion . . . . . 189
    - 1.1.5 Sortie de SQL\*Plus . . . . . 189
  - 1.2 Exécution des instructions . . . . . 190
    - 1.2.1 Gestion du buffer . . . . . 191
    - 1.2.2 Utilisation de scripts . . . . . 195
  - 1.3 Gestion de l'environnement SQL\*Plus . . . . . 198
- 2. Présentation des données . . . . . 202
  - 2.1 Gestion des variables . . . . . 202
  - 2.2 Présentation des résultats . . . . . 207
    - 2.2.1 Contrôle du déroulement des scripts . . . . . 207
    - 2.2.2 En-tête et pied de page . . . . . 207
    - 2.2.3 Rupture . . . . . 209
    - 2.2.4 Format de colonne . . . . . 209
    - 2.2.5 Calcul statistique . . . . . 211
    - 2.2.6 Annulation des déclarations . . . . . 212
  - 2.3 Environnement et traitement des erreurs . . . . . 212
    - 2.3.1 Statistiques de temps . . . . . 212
    - 2.3.2 Traitement des erreurs . . . . . 212
    - 2.3.3 Paramètres d'environnement . . . . . 213
  - 2.4 Création d'un rapport au format HTML . . . . . 217

3.	SQL Developer	220
3.1	Lancer SQL Developer	220
3.2	Créer une nouvelle connexion	222
3.3	Saisir des requêtes SQL	223
3.4	Mettre à jour les données	227
3.5	Travailler avec les objets d'un schéma	229
3.6	Exporter des données	233
3.7	Exporter des définitions	236

## Chapitre 4 PL/SQL

1.	Introduction	241
1.1	Qu'est-ce que le PL/SQL ?	241
1.2	Instructions SQL intégrées dans PL/SQL	242
1.3	Instructions spécifiques au PL/SQL	242
1.4	Le bloc PL/SQL	243
2.	Gestion des variables	244
2.1	Variables locales	244
2.2	Types prédéfinis	246
2.2.1	Types caractères	246
2.2.2	Types numériques	248
2.2.3	Types pour les grands objets	248
2.2.4	Autres types	249
2.2.5	Sous-types	249
2.3	Types définis par l'utilisateur	250
2.4	Collections et enregistrements	251
2.4.1	Les collections	251
2.4.2	Les enregistrements	254
2.5	Types dérivés	255
2.6	Variables définies dans un environnement extérieur à PL/SQL	256

- 2.7 Utilisation des variables ..... 257
  - 2.7.1 Affectation de valeur ..... 257
  - 2.7.2 Utilisation ..... 258
  - 2.7.3 Visibilité ..... 259
- 2.8 Variables structurées et instructions du DML ..... 259
- 3. Structures de contrôle ..... 262
  - 3.1 Traitements conditionnels ..... 263
  - 3.2 Traitements répétitifs ..... 266
- 4. Utilisation des curseurs ..... 269
  - 4.1 Définition ..... 269
  - 4.2 Étapes d'utilisation d'un curseur explicite ..... 269
    - 4.2.1 Déclaration ..... 269
    - 4.2.2 Ouverture ..... 270
    - 4.2.3 Traitement des lignes ..... 270
    - 4.2.4 Fermeture ..... 270
    - 4.2.5 Curseur FOR ..... 271
  - 4.3 Attributs d'un curseur ..... 273
  - 4.4 La variable ROWNUM ..... 275
  - 4.5 Modification des valeurs d'un curseur ..... 276
  - 4.6 Passage de paramètres ..... 277
- 5. Gestion des erreurs ..... 278
  - 5.1 Erreurs prédéfinies ..... 280
  - 5.2 Anomalies programme utilisateur ..... 282
  - 5.3 Erreurs Oracle ..... 284
  - 5.4 Portée des exceptions ..... 287
  - 5.5 Utilisation de raise\_application\_error ..... 288
- 6. Exemple récapitulatif ..... 290
  - 6.1 Énoncé du traitement ..... 290
  - 6.2 Exemple ..... 290
  - 6.3 Exécution par SQL\*Plus ..... 292

## Chapitre 5

### Objets de la base utilisant PL/SQL

1. Introduction . . . . .	295
2. Les déclencheurs de bases de données . . . . .	295
3. Les triggers sur des événements système ou utilisateur. . . . .	307
3.1 Les attributs . . . . .	308
3.2 Les événements système . . . . .	311
3.3 Les événements utilisateur. . . . .	312
4. Les modifications de triggers . . . . .	314
5. Les procédures stockées . . . . .	316
6. Les fonctions stockées . . . . .	318
7. Les packages . . . . .	323
7.1 En-tête. . . . .	324
7.2 Corps du package . . . . .	325
7.3 Utilisation . . . . .	327
7.4 Les curseurs . . . . .	329
8. Les transactions autonomes. . . . .	330
9. SQL dynamique . . . . .	334
9.1 EXECUTE IMMEDIATE . . . . .	335
9.2 OPEN FOR, FETCH et CLOSE . . . . .	339
9.2.1 Ouvrir un curseur (OPEN FOR) . . . . .	339
9.2.2 FETCH . . . . .	340
9.2.3 CLOSE . . . . .	342
9.3 Utilisation des curseurs dynamiques. . . . .	343
9.3.1 Amélioration des performances. . . . .	343
9.3.2 Passer le nom d'un objet. . . . .	343
9.3.3 Utiliser plusieurs fois le même argument . . . . .	344
9.3.4 Les attributs des curseurs. . . . .	344
9.3.5 Passer des valeurs NULL. . . . .	345
9.3.6 Utiliser les droits de l'utilisateur . . . . .	346

- 9.3.7 La directive de compilation  
      - RESTRICT\_REFERENCES ..... 347
    - 9.3.8 Éviter les verrous mortels. .... 348
  - 9.4 Le package DBMS\_SQL ..... 348
- 10. Collections et enregistrements ..... 350
  - 10.1 Référencer un élément d'une collection ..... 351
  - 10.2 Assigner une valeur et comparer des collections ..... 352
  - 10.3 Travailler avec les collections. .... 354
    - 10.3.1 Travailler avec les collections  
 de type NESTED TABLE ..... 354
    - 10.3.2 Travailler avec les tableaux ..... 356
  - 10.4 Manipuler les éléments des collections. .... 358
  - 10.5 Les méthodes. .... 359
    - 10.5.1 EXISTS ..... 360
    - 10.5.2 COUNT ..... 360
    - 10.5.3 LIMIT ..... 361
    - 10.5.4 FIRST, LAST ..... 361
    - 10.5.5 PRIOR, NEXT ..... 361
    - 10.5.6 EXTEND ..... 362
    - 10.5.7 TRIM ..... 363
    - 10.5.8 DELETE ..... 363
    - 10.5.9 COLLECT ..... 365
  - 10.6 Les exceptions ..... 366
- 11. La copie des données par blocs. .... 367
  - 11.1 FORALL ..... 369
    - 11.1.1 Limitations ..... 371
    - 11.1.2 Les transactions et la commande FORALL ..... 372
    - 11.1.3 Les clauses INDICES OF et VALUES OF ..... 373
  - 11.2 L'attribut %BULK\_ROWCOUNT ..... 373
  - 11.3 BULK COLLECT ..... 374
  - 11.4 LIMIT ..... 375
  - 11.5 Comparer les collections ..... 376
- 12. Fonctions et ensembles de lignes ..... 381

13. L'utilitaire Wrap .....	383
14. DBMS_OUTPUT .....	384
14.1 ENABLE .....	385
14.2 DISABLE .....	385
14.3 PUT et PUT_LINE .....	386
14.4 NEW_LINE .....	386
14.5 GET_LINE et GET_LINES .....	387
15. Le package UTL_FILE .....	388
15.1 FOPEN, FOPEN_NCHAR .....	389
15.2 IS_OPEN .....	391
15.3 FCLOSE .....	391
15.4 FCLOSE_ALL .....	392
15.5 GET_LINE, GET_LINE_NCHAR, GET_RAW .....	392
15.6 PUT, PUT_NCHAR, PUT_RAW .....	393
15.7 NEW_LINE .....	394
15.8 PUT_LINE, PUT_LINE_NCHAR .....	394
15.9 PUTF, PUTF_NCHAR .....	395
15.10 FFLUSH .....	396
15.11 FSEEK, FGETPOS .....	397
15.12 FREMOVE, FCOPY, FRENAME .....	397
15.13 FGETATTR .....	398
15.14 Les exceptions .....	398
16. Le package DBMS_LOB .....	400
16.1 Les constantes .....	401
16.2 APPEND .....	401
16.3 CLOSE .....	401
16.4 COMPARE .....	402
16.5 COPY .....	402
16.6 CREATETEMPORARY, FREETEMPORARY, ISTEMPORARY .....	403
16.7 ERASE .....	404
16.8 FILEOPEN, FILECLOSE, FILECLOSEALL et ISOPEN .....	405
16.9 FILEEXISTS, FILEISOPEN .....	405

16.10 FILEGETNAME ..... 406  
16.11 FRAGMENT\_DELETE, FRAGMENT\_INSERT,  
FRAGMENT\_MOVE, FRAGMENT\_REPLACE..... 406  
16.12 GETLENGTH, GETCHUNKSIZE..... 407  
16.13 INSTR..... 408  
16.14 LOADFROMFILE, LOADBLOBFROMFILE,  
LOADCLOBFROMFILE..... 409  
16.15 OPEN..... 412  
16.16 READ..... 412  
16.17 SUBSTR ..... 412  
16.18 TRIM..... 413  
16.19 WRITE, WRITEAPPEND..... 413  
16.20 Les exceptions..... 414

**Chapitre 6**  
**Java**

1. Introduction ..... 417  
2. Chargement des procédures stockées ..... 418  
2.1 Généralités ..... 419  
2.2 Les droits d'utilisation ..... 420  
2.3 L'utilitaire loadjava ..... 421  
2.4 L'utilitaire dropjava..... 424  
2.5 L'accès aux données ..... 424  
2.5.1 JDBC ..... 425  
2.5.2 SQLJ..... 425  
3. Publication des procédures stockées ..... 426  
3.1 Correspondance des types de données ..... 427  
3.2 Création d'une fonction Java ou d'une procédure Java ..... 428  
3.3 CREATE JAVA ..... 431

4.	Utilisation des procédures stockées. . . . .	432
4.1	Appel d'une procédure Java depuis SQL*Plus. . . . .	432
4.2	Appel d'une procédure Java depuis un déclencheur de base de données . . . . .	436
4.3	Appel d'une procédure Java depuis une instruction SQL DML ou un bloc PL/SQL . . . . .	437

## Chapitre 7

### Le parseur XML

1.	Introduction . . . . .	441
2.	Lire un fichier XML . . . . .	443
3.	Appliquer une feuille de style à un document XML . . . . .	446
4.	XSU . . . . .	448
4.1	Génération de code XML avec DBMS_XMLQuery . . . . .	448
4.1.1	Génération de code XML depuis une requête. . . . .	449
4.1.2	Modifier les noms des balises ROW et ROWSET . . . . .	451
4.1.3	Limiter le nombre de lignes . . . . .	452
4.1.4	Les feuilles de style . . . . .	454
4.2	Les requêtes paramétrées . . . . .	454
4.3	Stocker les informations au format XML avec DBMS_XMLSave . . . . .	456
4.3.1	Ajouter des données. . . . .	457
4.3.2	Mettre à jour des données . . . . .	460
4.3.3	Supprimer des données . . . . .	463

**Chapitre 8**  
**Application Express**

- 1. Introduction ..... 465
- 2. Activer Oracle Application Express..... 466
- 3. Créer un espace de travail ..... 468
- 4. Développer une application ..... 473
  - 4.1 Se connecter à l'espace de travail ..... 473
  - 4.2 Créer les objets de l'application ..... 475
  - 4.3 Créer l'application..... 483
  - 4.4 Tester l'application ..... 490
  - 4.5 Personnaliser l'application ..... 494
    - 4.5.1 Modifier la présentation des pages ..... 494
    - 4.5.2 Créer et utiliser des listes de valeurs ..... 499
    - 4.5.3 Ajouter des contrôles sur les données saisies ..... 508
  
- Index ..... 515

## Chapitre 5

# Objets de la base utilisant PL/SQL

### 1. Introduction

En plus des blocs PL/SQL anonymes utilisés par SQL\*Plus ou par les outils de développement (Oracle\*Forms, Oracle\*Reports...), on peut utiliser le PL/SQL dans des objets de la base, comme les procédures stockées (PROCEDURE, FUNCTION, PACKAGE) et les déclencheurs de bases de données.

### 2. Les déclencheurs de bases de données

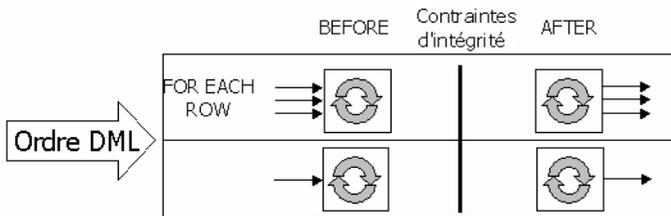
Un déclencheur ou trigger est un bloc PL/SQL associé à une table. Ce bloc s'exécutera lorsqu'une instruction du DML (INSERT, UPDATE, DELETE) sera demandée sur la table.

Le bloc PL/SQL qui constitue le trigger peut être exécuté avant ou après la mise à jour et donc avant ou après vérification des contraintes d'intégrité.

Les triggers offrent une solution procédurale pour définir des contraintes complexes ou qui prennent en compte des données issues de plusieurs lignes ou de plusieurs tables. Par exemple, pour garantir le fait qu'un client ne peut pas avoir plus de deux commandes non payées. Toutefois, les triggers ne doivent pas être utilisés lorsqu'il est possible de mettre en place une contrainte d'intégrité. En effet, les contraintes d'intégrité étant définies au niveau de la table et faisant partie de la structure de la table, la vérification du respect de ces contraintes est beaucoup plus rapide.

De plus, les contraintes d'intégrité garantissent que toutes les lignes contenues dans la table respectent ces contraintes, tandis que les triggers ne prennent pas en compte les données déjà présentes dans la table lorsqu'ils sont définis.

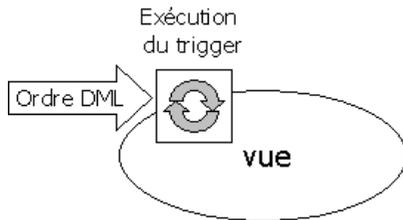
Le bloc PL/SQL associé au trigger peut être exécuté pour chaque ligne affectée par l'ordre DML (option FOR EACH ROW), ou bien une seule fois pour chaque commande DML exécutée (option par défaut).



*Exécution avant ou après vérification des contraintes d'intégrité pour chaque ligne ou chaque ordre.*

Dans les triggers BEFORE et FOR EACH ROW, il est possible de modifier les données qui vont être insérées dans la table pour qu'elles respectent les contraintes d'intégrité. Il est également possible de faire des requêtes de type SELECT sur la table sur laquelle porte l'ordre DML uniquement dans le cadre d'un trigger BEFORE INSERT. Toutes ces opérations sont impossibles dans les triggers AFTER car après vérification des contraintes d'intégrité, il n'est pas possible de modifier les données et comme la modification (ou ajout, ou suppression) de la ligne n'est pas terminée, il n'est pas possible d'effectuer des SELECT sur la table.

Il est également possible de poser des triggers sur les vues (VIEW) afin d'intercepter les ordres DML que l'on peut y exécuter. Ces triggers permettent de maîtriser toutes les opérations qui sont effectuées sur les vues et pour l'utilisateur final, la vue est en tout point similaire à une table puisqu'il peut y faire les opérations INSERT, UPDATE et DELETE. Ces triggers sont de type INSTEAD OF, c'est-à-dire que leur exécution va remplacer celle de la commande DML à laquelle ils sont associés. Ce type de trigger n'est définissable que sur les vues, et seul ce type de trigger peut être mis en place sur les vues.



*Principe de fonctionnement des triggers instead of*

### Syntaxe

```
CREATE [OR REPLACE] TRIGGER nom_trigger  
{BEFORE/AFTER/INSTEAD OF}  
{INSERT/UPDATE[OF col,...]/DELETE}  
ON nom_table [FOR EACH ROW]  
[FOLLOWS nom_autre_trigger[,...]]  
[ENABLE/DISABLE]  
[WHEN (condition)]  
Bloc PL/SQL
```

OR REPLACE

Remplace la description du trigger s'il existe déjà.

BEFORE

Le bloc PL/SQL est exécuté avant la vérification des contraintes de tables et la mise à jour des données dans la table.

AFTER

Le bloc PL/SQL est exécuté après la mise à jour des données dans la table.

INSTEAD OF

Le bloc PL/SQL qui suit remplace le traitement standard associé à l'instruction qui a déclenché le trigger (pour une vue uniquement).

INSERT/UPDATE [OF col, ...]/DELETE

Instruction associée au déclenchement du trigger. Plusieurs instructions peuvent déclencher le même trigger. Elles sont combinées par l'opérateur OR.

FOR EACH ROW

Le trigger s'exécute pour chaque ligne traitée par l'instruction associée.

FOLLOWS nom\_autre\_trigger[,...]

Oracle permet de définir plusieurs triggers pour la même table et le même événement. Dans ce cas, l'ordre relatif de déclenchement de ces triggers est indéterminé. Si l'ordre de déclenchement de ces triggers est important pour votre application, vous pouvez utiliser la clause FOLLOWS apparue en version 11. Cette clause permet d'indiquer que le trigger doit être déclenché après les triggers mentionnés.

ENABLE/DISABLE

Cette clause permet d'indiquer si le trigger est actif ou non dès sa création ; par défaut, un trigger nouvellement créé est actif. Créer un trigger désactivé permet de vérifier qu'il se compile correctement avant de le mettre réellement en service. Un trigger créé désactivé peut ensuite être activé par un ordre ALTER TRIGGER ... ENABLE.

WHEN (condition)

La condition donnée doit être vérifiée pour que le code s'exécute.

Les données de la table à laquelle est associé le trigger sont inaccessibles depuis les instructions du bloc. Seule la ligne en cours de modification est accessible à l'aide de deux variables RECORD, OLD et NEW, qui reprennent la structure de la TABLE ou de la VIEW associée. Ces variables peuvent être utilisées dans la clause WHEN du trigger ou dans le bloc d'instructions. Dans ce dernier cas, elles sont référencées comme des variables hôtes avec le préfixe ":" (:OLD.nom\_champs, :NEW.nom\_champs).

Le terme OLD permet de connaître la ligne en cours de suppression dans un trigger DELETE ou la ligne avant modification dans un trigger UPDATE. Le terme NEW permet de connaître la nouvelle ligne insérée dans un trigger INSERT ou la ligne après modification dans un trigger UPDATE.

Les termes OLD et NEW sont fixés par défaut et il est possible d'utiliser d'autres termes en précisant la clause REFERENCING OLD AS nouveau\_nom NEW AS nouveau\_nom. Cette clause prend place juste avant la clause FOR EACH ROW (si elle existe) dans la définition du trigger.

### Exemple

Exécution d'un bloc PL/SQL avant une suppression dans la table CLIENTS du user FLORIAN :

```
CREATE TRIGGER avant_sup_cli
  BEFORE DELETE
  ON FLORIAN.CLIENTS
  DECLARE
    ...
  BEGIN
    ...
  END ;
```

Exécution d'un bloc PL/SQL après mise à jour de chaque ligne de la table ARTICLES si l'ancien prix est supérieur au nouveau :

```
create or replace trigger post_majprix
  after update of prix
  on articles
  for each row
  when (old.prix > new.prix)
  declare
    ...
  begin
    ...
  end;
```

Pour chaque commande on souhaite connaître le nom de l'utilisateur Oracle qui a réalisé la saisie. La première étape consiste à ajouter une nouvelle colonne à la table des commandes. Cette colonne doit accepter la valeur NULL car pour les lignes de commande existantes, le nom de l'utilisateur Oracle est inconnu.

Modification de la table des commandes :

```
SQL> alter table commandes
  2   add (utilisateur varchar2(30));

Table modifiée.

SQL>
```

Dans le trigger, le nom de la nouvelle ligne est changé, il s'exécute avant vérification des contraintes d'intégrité pour chaque ligne insérée dans la table commandes.

Définition du trigger :

```
SQL> create or replace trigger bf_ins_commandes
  2   before insert
  3   on commandes
  4   referencing new as nouvelle
  5   for each row
  6   begin
  7     select user into :nouvelle.utilisateur from dual;
  8   end;
  9   /
```

Déclencheur créé.

```
SQL>
```

On souhaite connaître le nombre de commandes saisies par utilisateur Oracle. Pour éviter d'écrire une requête qui parcourt la totalité de la table des commandes, ce qui peut être très lourd, une table de statistiques va être alimentée par le trigger.

Création de la table de statistiques :

```
SQL> create table stat_util(
  2   utilisateur varchar2(30),
  3   nombre integer);
```

Table créée.

```
SQL>
```

Le trigger doit s'assurer que l'utilisateur existe dans la table des statistiques, et s'il n'est pas déjà présent alors il faut le créer. Le trigger s'exécute après chaque insertion de ligne, pour des raisons d'optimisation en cas de violation des contraintes d'intégrité.