

Christian Soutou

Couvre Oracle 12c

SQL

pour

Oracle

7^e édition

Applications avec Java, PHP et XML
Optimisation des requêtes et schémas

Avec 50 exercices corrigés

EYROLLES

Christian Soutou est maître de conférences à l'université Toulouse Jean-Jaurès et consultant indépendant. Rattaché au département Réseaux et Télécoms de l'IUT de Blagnac, il intervient autour des technologies de l'information en DUT, licence et master professionnels, ainsi que pour le compte de la société Orsys. Il est également l'auteur d'ouvrages sur SQL Server, MySQL, UML et les bases de données, tous parus aux éditions Eyrolles.

Apprendre SQL par l'exemple

Tout particulièrement destiné aux débutants et aux étudiants, cet ouvrage permet d'acquérir les notions essentielles d'Oracle, leader des systèmes de gestion de bases de données. Concis et de difficulté progressive, il est émaillé de nombreux exemples et de 50 exercices corrigés qui illustrent tous les aspects fondamentaux de SQL. Couvrant les versions 9i à 12c d'Oracle, il permet de se familiariser avec ses principales fonctionnalités, ainsi qu'avec les API les plus utilisées (JDBC, PHP et XML DB). Ce livre consacre également un chapitre entier à l'optimisation des requêtes et des schémas relationnels, en étudiant l'optimiseur, les statistiques, la mesure des performances et l'emploi de la boîte à outils : contraintes, index, tables organisées en index, partitionnement, vues matérialisées et dénormalisation. Mise à jour et augmentée, cette septième édition actualise la partie XML DB et présente l'architecture multitenant de la version 12c.

À qui s'adresse cet ouvrage ?

- À tous ceux qui souhaitent s'initier à SQL, à Oracle ou à la gestion de bases de données
- Aux développeurs C, C++, Java, PHP et XML qui souhaitent stocker leurs données

Installez vous-même Oracle !

Les compléments web de cet ouvrage décrivent en détail les procédures d'installation des différentes versions d'Oracle, de la 9i à la 12c (éditions *Express* et *Enterprise*). Ces versions peuvent être téléchargées gratuitement sur le site d'Oracle : destinées à des fins non commerciales, elles sont complètes et sans limitation de durée.

Au sommaire

Partie I : SQL de base. Définition des données. Manipulation des données. Évolution d'un schéma. Interrogation des données. Contrôle des données. **Partie II : PL/SQL.** Bases du PL/SQL. Programmation avancée. **Partie III : SQL avancé.** Le précompilateur Pro*C/C++. L'interface JDBC. Oracle et PHP. Oracle XML DB. Optimisation.



Sur le site www.editions-eyrolles.com

- Téléchargez le code source des exemples et le corrigé des exercices
- Consultez les mises à jour et les compléments
- Dialoguez avec l'auteur

Code éditeur : G14156
ISBN : 978-2-212-14156-6

SQL pour Oracle

DU MÊME AUTEUR

C. SOUTOU, F. BROUARD, N. SOUQUET et D. BARBARIN. – **SQL Server 2014**.
N°13592, 2015, 890 pages.

C. SOUTOU. – **Programmer avec MySQL (3^e édition)**.
N°13719, 2013, 520 pages.

C. SOUTOU. – **Modélisation de bases de données (3^e édition)**.
N°14206, 2015, 352 pages. *À paraître*.

AUTOUR D'ORACLE ET DE SQL

R. BIZOI – **Oracle 12c – Administration**.
N°14056, 2014, 564 pages.

R. BIZOI – **Oracle 12c – Sauvegarde et restauration**.
N°14057, 2014, 336 pages.

R. BIZOI – **SQL pour Oracle 12c**.
N°14054, 2014, 416 pages.

R. BIZOI – **PL/SQL pour Oracle 12c**.
N°14055, 2014, 340 pages.

C. PIERRE DE GEYER et G. PONÇON – **Mémento PHP et SQL (3^e édition)**.
N°13602, 2014, 14 pages.

R. BIZOI – **Oracle 11g – Administration**.
N°12899, 2011, 600 pages.

R. BIZOI – **Oracle 11g – Sauvegarde et restauration**.
N°12899, 2011, 432 pages.

G. BRIARD – **Oracle 10g sous Windows**.
N°11707, 2006, 846 pages.

R. BIZOI – **SQL pour Oracle 10g**.
N°12055, 2006, 650 pages.

G. BRIARD – **Oracle 10g sous Windows**.
N°11707, 2006, 846 pages.

G. BRIARD – **Oracle9i sous Linux**.
N°11337, 2003, 894 pages.

Christian Soutou

SQL pour Oracle

7^e édition

**Applications avec Java, PHP et XML
Optimisation des requêtes et schémas**

Avec 50 exercices corrigés

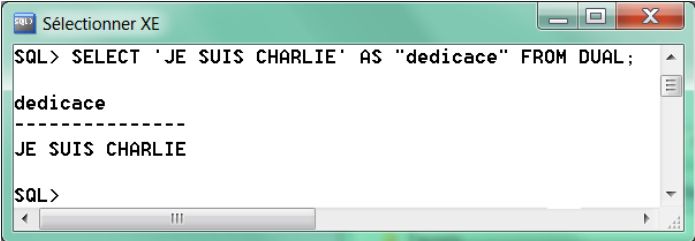
EYROLLES

The logo for EYROLLES, featuring the word "EYROLLES" in a bold, sans-serif font, centered above a horizontal line with a small circle in the middle.

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.
© Groupe Eyrolles, 2004-2015, ISBN : 978-2-212-14156-6

Si Oracle était doué d'écriture, il penserait certainement aux journalistes et aux autres victimes qui ont perdu la vie au cours des attentats de Paris en janvier 2015.



```
Sélectionner XE
SQL> SELECT 'JE SUIS CHARLIE' AS "dedicace" FROM DUAL;
dedicace
-----
JE SUIS CHARLIE
SQL>
```


Avant-propos

Nombre d'ouvrages traitent de SQL et d'Oracle ; certains résultent d'une traduction hasardeuse et sans vocation pédagogique, d'autres ressemblent à des annuaires téléphoniques. Les survivants, bien qu'intéressants, ne sont quant à eux plus vraiment à jour.

Ce livre a été rédigé avec une volonté de concision et de progression dans sa démarche ; il est illustré par ailleurs de nombreux exemples et figures. Bien que notre source principale d'informations fût la documentation en ligne d'Oracle, l'ouvrage ne constitue pas, à mon sens, un simple condensé de commandes SQL. Chaque notion importante est introduite par un exemple facile et démonstratif (du moins je l'espère). À la fin de chaque chapitre, des exercices vous permettront de tester vos connaissances.

Depuis quelques années, la documentation d'Oracle représente des centaines d'ouvrages au format HTML ou PDF (soit plusieurs dizaines de milliers de pages) ! Ainsi, il est vain de vouloir expliquer tous les concepts, même si cet ouvrage ressemblait à un annuaire. J'ai tenté d'extraire les aspects fondamentaux sous la forme d'une synthèse. Ce livre résulte de mon expérience d'enseignement dans des cursus d'informatique à vocation professionnelle (IUT, master professionnel et interentreprise).

Cet ouvrage s'adresse principalement aux novices désireux de découvrir SQL et de programmer sous Oracle.

- Les étudiants trouveront des exemples pédagogiques pour chaque concept abordé, ainsi que des exercices thématiques.
- Les développeurs C, C++, PHP ou Java découvriront des moyens de stocker leurs données.
- Les professionnels connaissant déjà Oracle seront peut-être intéressés par certaines nouveautés décrites dans cet ouvrage.

Les fonctionnalités de la version 11g ont été prises en compte lors de la troisième édition de cet ouvrage. Certains mécanismes d'optimisation (index, *clusters*, partitionnement, tables organisées en index, vues matérialisées et dénormalisation) sont apparus lors de la quatrième édition en même temps que quelques nouveautés SQL (pivots, transpositions, requêtes *pipeline*, CTE et récursivité). La cinquième édition enrichissait l'intégration avec Java (connexion à une base MySQL, *Data Sources* et *RowSets*) et PHP (API PDO : *PHP Data Objects*). La sixième édition présentait l'outil *SQL Data Modeler*. Celle-ci inclut des nouveautés de la version 12c et actualise principalement la technologie XML DB.

Par ailleurs, plusieurs compléments qui concernent des usages d'Oracle moins courants sont disponibles en téléchargement sur la fiche de l'ouvrage (à l'adresse www.editions-eyrolles.com) :

- l'installation de différentes versions (complément 1 : Installation des versions 9i à 12c) ;
- la technologie SQLJ (complément 2 : L'approche SQLJ) ;
- les procédures externes (complément 3 : Procédures stockées et externes) ;
- les fonctions PL/SQL pour construire des pages HTML (complément 4 : PL/SQL Web Toolkit et PL/SQL Server Pages).

Guide de lecture

Ce livre s'organise autour de trois parties distinctes mais complémentaires. La première intéressera le lecteur novice en la matière, car elle concerne les instructions SQL et les notions de base d'Oracle. La deuxième partie décrit la programmation avec le langage procédural d'Oracle PL/SQL. La troisième partie attirera l'attention des programmeurs qui envisagent d'utiliser Oracle tout en programmant avec des langages évolués (C, C++, PHP ou Java) ou via des interfaces Web.

Première partie : SQL de base

Cette partie présente les différents aspects du langage SQL d'Oracle en étudiant en détail les instructions élémentaires. À partir d'exemples simples et progressifs, nous expliquons notamment comment déclarer, manipuler, faire évoluer et interroger des tables avec leurs différentes caractéristiques et éléments associés (contraintes, index, vues, séquences). Nous étudions aussi SQL dans un contexte multi-utilisateur (droits d'accès), et au niveau du dictionnaire de données.

Deuxième partie : PL/SQL

Cette partie décrit les caractéristiques du langage procédural PL/SQL d'Oracle. Le chapitre 6 aborde des éléments de base (structure d'un programme, variables, structures de contrôle, interactions avec la base, transactions). Le chapitre 7 traite des sous-programmes, des curseurs, de la gestion des exceptions, des déclencheurs et de l'utilisation du SQL dynamique.

Troisième partie : SQL avancé

Cette partie intéressera les programmeurs qui envisagent d'exploiter une base Oracle en utilisant un langage de troisième ou quatrième génération (C, C++ ou Java), ou en employant une interface Web. Le chapitre 8 est consacré à l'étude des mécanismes de base du précompilateur d'Oracle Pro*C/C++. Le chapitre 9 présente les principales fonctionnalités de l'API JDBC.

Le chapitre 10 traite des deux principales API disponibles avec le langage PHP (OCI8 et PDO). Le chapitre 11 présente les fonctionnalités de XML DB et l'environnement *XML DB Repository*. Enfin, le chapitre 12 est dédié à l'optimisation des requêtes et des schémas relationnels.

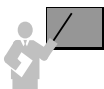
Conventions d'écriture et pictogrammes

La police *courrier* est utilisée pour souligner les instructions SQL, noms de types, tables, contraintes, etc. (exemple : `SELECT nom FROM Pilote`).

Les majuscules sont employées pour les directives SQL, et les minuscules pour les autres éléments. Les noms des tables, index, vues, fonctions, procédures, etc., sont précédés d'une majuscule (exemple : la table `CompagnieAerienne` contient la colonne `nomComp`).

Les termes d'Oracle (bien souvent traduits littéralement de l'anglais) sont notés en italique (exemple : *row*, *trigger*, *table*, *column*, etc.).

Dans une instruction SQL, les symboles { et } désignent une liste d'éléments, et le symbole | un choix (par exemple, `CREATE {TABLE | VIEW}` exprime deux instructions possibles : `CREATE TABLE` ou `CREATE VIEW`). Les signes [et] désignent le caractère facultatif d'une option (par exemple, `CREATE TABLE Avion(...) [TABLESPACE USERS]` exprime deux écritures possibles : `CREATE TABLE Avion(...) TABLESPACE USERS` et `CREATE TABLE Avion(...)`).



Ce pictogramme introduit une définition, un concept ou une remarque importante. Il apparaît soit dans une partie théorique, soit dans une partie technique, pour souligner des instructions importantes ou la marche à suivre avec SQL.



Ce pictogramme annonce soit une impossibilité de mise en œuvre d'un concept, soit une mise en garde. Il est principalement utilisé dans la partie consacrée à SQL.



Ce pictogramme indique une astuce ou un conseil personnel.



Ce pictogramme indique une commande ou option disponible uniquement à partir de la version 12c.

Contact avec l'auteur et site Web

Si vous avez des remarques à formuler sur le contenu de cet ouvrage, n'hésitez pas à m'écrire (christian.soutou@gmail.com). Vous trouverez sur le site d'accompagnement, accessible par www.editions-eyrolles.com, les compléments et errata, ainsi que le code de tous les exemples et les exercices corrigés.

Table des matières

Introduction	1
SQL, une norme, un succès	1
Modèle de données	2
Tables et données	2
Les clés	3
Oracle	3
Un peu d'histoire	4
Rachat de Sun (et de MySQL)	5
Offre du moment	6
Notion de schéma	8
Accès à Oracle depuis Windows	9
Détail d'un numéro de version	9
Les clients SQL	10
SQL*Plus	10
SQL Developer	11
SQL Data Modeler	12
Premiers pas	13
Variables d'environnement	15
À propos des accents et jeux de caractères	16
Partie I SQL de base	19
1 Définition des données	21
Tables relationnelles	21
Création d'une table (CREATE TABLE)	21
Casse et commentaires	22
Premier exemple	23
Contraintes de colonnes	23
Conventions recommandées	24
Types des colonnes	26
Structure d'une table (DESC)	31
Commentaires stockés (COMMENT)	31
Noms des objets	32
Utilisation de SQL Developer Data Modeler	33
Suppression des tables	37

2	Manipulation des données	43
	Insertions d'enregistrements (INSERT)	43
	Syntaxe	43
	Renseigner ou pas toutes les colonnes	44
	Ne pas respecter des contraintes	45
	Dates/heures	46
	Caractères Unicode	49
	Données LOB	50
	Séquences	51
	Création d'une séquence (CREATE SEQUENCE)	51
	Manipulation d'une séquence	54
	Utilisation d'une séquence dans un DEFAULT	56
	Modification d'une séquence (ALTER SEQUENCE)	56
	Visualisation d'une séquence	57
	Suppression d'une séquence (DROP SEQUENCE)	58
	Colonnes auto-incrémentées	58
	Modifications de valeurs	59
	Syntaxe (UPDATE)	60
	Modification d'une ligne	60
	Modification de plusieurs lignes	60
	Ne pas respecter des contraintes	61
	Dates et intervalles	62
	Suppressions d'enregistrements	66
	Instruction DELETE	66
	Instruction TRUNCATE	67
	Intégrité référentielle	68
	Cohérences	68
	Contraintes côté « père »	69
	Contraintes côté « fils »	69
	Clés composites et nulles	70
	Cohérence du fils vers le père	71
	Cohérence du père vers le fils	71
	En résumé	72
3	Évolution d'un schéma	77
	Renommer une table (RENAME)	77
	Modifications structurelles (ALTER TABLE)	78
	Ajouter des colonnes	78
	Renommer des colonnes	79
	Modifier le type des colonnes	79
	Supprimer des colonnes	80
	Colonnes UNUSED	80
	Colonne virtuelle	81
	Colonnes invisibles	83

Modifications comportementales	84
Ajout de contraintes	84
Suppression de contraintes	85
Désactivation de contraintes	87
Réactivation de contraintes	89
Contraintes différées	92
Directives DEFERRABLE et INITIALLY	92
Instructions SET CONSTRAINT	94
Instruction ALTER SESSION SET CONSTRAINTS	94
Directives VALIDATE et NOVALIDATE	94
Directive MODIFY CONSTRAINT	96
4 Interrogation des données	101
Généralités	101
Syntaxe (SELECT)	102
Pseudo-table DUAL	102
Projection (éléments du SELECT)	103
Extraction de toutes les colonnes	104
Extraction de certaines colonnes	105
Alias	105
Duplicatas	106
Expressions	106
Ordonnancement	107
Substitutions conditionnelles	108
Pseudo-colonne ROWID	108
Pseudo-colonne ROWNUM	109
Insertion multiligne	109
Limitation du nombre de lignes	110
Restriction (WHERE)	111
Opérateurs de comparaison	112
Opérateurs logiques	113
Opérateurs intégrés	114
Fonctions	115
Caractères	115
Numériques	119
Valeurs spéciales pour les flottants	120
Fonctions pour les flottants	120
Dates	124
Conversions	125
Autres fonctions	126
Regroupements	127
Fonctions de groupe	128
Étude du GROUP BY et HAVING	129

Opérateurs ensemblistes	132
Restrictions	132
Exemple	133
Opérateur INTERSECT	133
Opérateurs UNION et UNION ALL	134
Opérateur MINUS	134
Ordonner les résultats	135
Produit cartésien	136
Bilan	137
Sous-interrogations dans la clause FROM	138
Jointures	140
Classification	140
Jointure relationnelle	141
Jointures SQL2	141
Types de jointures	142
Équijointure	142
Autojointure	144
Inéquijointure	145
Jointures externes	146
Jointures procédurales	151
Jointures mixtes	155
Sous-interrogations synchronisées	155
Autres directives SQL2	157
Division	159
Définition	160
Classification	160
Division inexacte en SQL	161
Division exacte en SQL	162
Requêtes hiérarchiques	162
Point de départ du parcours (START WITH)	163
Parcours de l'arbre (CONNECT BY PRIOR)	163
Indentation	164
Élagage de l'arbre (WHERE et PRIOR)	165
Jointures	167
Ordonnancement	167
Extraction de chemins	168
Extraction d'un élément	169
Nature d'un élément	169
Éviter un cycle	170
Mises à jour conditionnées (fusions)	172
Syntaxe (MERGE)	172
Exemple	173
Suppressions dans la table cible	173
Exemple	174

Expressions régulières	175
Quelques exemples	177
Fonction REGEXP_LIKE	177
Fonction REGEXP_REPLACE	180
Fonction REGEXP_INSTR	181
Fonction REGEXP_SUBSTR	183
Sous-expressions	184
Extractions diverses	185
Directive WITH	185
Fonction WIDTH_BUCKET	187
Récursivité avec WITH (CTE)	188
Pivots (PIVOT)	197
Transpositions (UNPIVOT)	201
Fonction LISTAGG	203
5 Contrôle des données	209
Les tablespaces	210
Indépendance logique/physique	210
Tablespaces déjà livrés	210
Création d'un tablespace	212
Gestion des utilisateurs	212
Classification	213
Création d'un utilisateur (CREATE USER)	213
Modification d'un utilisateur (ALTER USER)	215
Suppression d'un utilisateur (DROP USER)	216
Profils	216
Privilèges	219
Privilèges système	219
Privilèges objets	221
Privilèges prédéfinis	224
Rôles	225
Création d'un rôle (CREATE ROLE)	226
Rôles prédéfinis	228
Révocation d'un rôle	228
Activation d'un rôle (SET ROLE)	229
Modification d'un rôle (ALTER ROLE)	230
Suppression d'un rôle (DROP ROLE)	231
Vues	231
Création d'une vue (CREATE VIEW)	232
Classification	234
Vues monotables	234
Vues complexes	239
Autres utilisations de vues	242
Transmission de droits	246

Modification d'une vue (ALTER VIEW)	246
Suppression d'une vue (DROP VIEW)	246
Synonymes	247
Création d'un synonyme (CREATE SYNONYM)	247
Transmission de droits	249
Suppression d'un synonyme (DROP SYNONYM)	249
Dictionnaire des données	249
Constitution	250
Classification des vues	250
Démarche à suivre	251
Principales vues	253
Objets d'un schéma.	255
Structure d'une table	255
Recherche des contraintes d'une table	256
Composition des contraintes d'une table	256
Détails des contraintes référentielles	256
Recherche du code source d'un sous-programme	257
Recherche des utilisateurs d'une base de données	258
Rôles reçus	258
Le multitenant	259
Les consoles d'administration	262
Enterprise Manager Database Express	262
SQL Developer	264

Partie II PL/SQL

271

6 Bases du PL/SQL

273

Généralités	273
Environnement client-serveur	273
Avantages	274
Structure d'un programme.	274
Portée des objets	275
Jeu de caractères	276
Identificateurs	276
Commentaires	277
Variables	277
Variables scalaires	278
Affectations	278
Restrictions	279
Variables %TYPE	279
Variables %ROWTYPE	280
Variables RECORD	281
Variables tableaux (type TABLE)	282

Résolution de noms	284
Opérateurs	284
Variables de substitution	285
Variables de session	286
Conventions recommandées	286
Types de données PL/SQL	287
Types prédéfinis	287
Sous-types	287
Le sous-type SIMPLE_INTEGER	288
Les sous-types flottants	289
Variable de type séquence	289
Conversions de types	290
Structures de contrôles	290
Structures conditionnelles	290
Structures répétitives	293
La directive CONTINUE	297
Interactions avec la base	298
Extraire des données	298
Manipuler des données	300
Curseurs implicites	302
Paquetage DBMS_OUTPUT	303
Transactions	306
Caractéristiques	306
Début et fin d'une transaction	307
Contrôle des transactions	308
Niveaux d'isolation	308
Le problème du verrou mortel (deadlock)	311
Verrouillage manuel	313
Transactions imbriquées	314
Où placer les transactions ?	314
7 Programmation avancée	317
Sous-programmes	317
Généralités	317
Procédures cataloguées	318
Fonctions cataloguées	319
Codage d'un sous-programme PL/SQL	320
Exemples	320
Compilation	323
Appels	323
À propos des paramètres	325
Récursivité	326
Sous-programmes imbriqués	326
Recompilation d'un sous-programme	328

Destruction d'un sous-programme	328
Paquetages (packages)	328
Généralités	328
Spécification	329
Compilation	330
Implémentation	330
Appel	331
Surcharge	331
Recompilation	331
Destruction d'un paquetage	331
Comment retourner une table ?	332
 Curseurs	332
Généralités	333
Instructions	333
Parcours d'un curseur	334
Utilisation de structures (%ROWTYPE)	335
Boucle FOR (gestion semi-automatique)	336
Utilisation de tableaux (type TABLE)	337
Utilisation de LIMIT et BULK COLLECT	338
Paramètres d'un curseur	339
Accès concurrents (FOR UPDATE et CURRENT OF)	340
Variables curseurs (REF CURSOR)	341
Fonctions table pipelined	343
 Exceptions	345
Généralités	345
Exception interne prédéfinie	347
Exception utilisateur	351
Utilisation du curseur implicite	353
Exception interne non prédéfinie	354
Propagation d'une exception	355
Procédure RAISE_APPLICATION ERROR	357
 Déclencheurs	358
À quoi sert un déclencheur ?	358
Généralités	359
Mécanisme général	359
Syntaxe	360
Déclencheurs LMD	361
Transactions autonomes	373
Déclencheurs LDD	374
Déclencheurs d'instances	374
Appels de sous-programmes	375
Gestion des déclencheurs	376
Ordre d'exécution	377
Tables mutantes	377

	Activation et désactivation	378
	Ordre d'exécution (FOLLOWS)	378
	Déclencheur composé	379
	Résolution au problème des tables mutantes	381
	SQL dynamique	382
	Classification	383
	Utilisation de EXECUTE IMMEDIATE	384
	Utilisation d'une variable curseur	385
	Nouveautés de la version 12c	386
Partie III	SQL avancé	393
8	Le précompilateur Pro*C/C++	395
	Généralités	395
	Ordres SQL intégrés	395
	Variables	396
	Variable indicatrice	397
	Cas du VARCHAR	398
	Zone de communication (SQLCA)	398
	Connexion à une base	399
	Gestion des exceptions	399
	Transactions	400
	Extraction d'un enregistrement	400
	Mises à jour	402
	Utilisation de curseurs	402
	Variables scalaires	402
	Variables tableaux	403
	Utilisation de Microsoft Visual C++	405
9	L'interface JDBC	407
	Généralités	407
	Classification des pilotes (drivers)	408
	Les paquetages	409
	Structure d'un programme	410
	Variables d'environnement	411
	Test de votre configuration	412
	Connexion à une base	412
	Base Access	413
	Base Oracle	414
	Base MySQL	416
	Déconnexion	417
	Interface Connection	417
	Sources de données	417

États d'une connexion	418
Interfaces disponibles	418
Méthodes génériques pour les paramètres	419
États simples (interface Statement)	419
Méthodes à utiliser	420
Correspondances de types	421
Interactions avec la base	422
Suppression de données	422
Ajout d'enregistrements	423
Modification d'enregistrements	423
Extraction de données	423
Curseurs statiques	424
Curseurs navigables	425
Curseurs modifiables	429
Suppressions	431
Modifications	432
Insertions	432
Restrictions	433
Ensembles de lignes (RowSet)	434
RowSet sans connexion	435
RowSet avec ResultSet	435
RowSet pour XML	436
Mises à jour d'un RowSet	437
Notifications pour un RowSet	437
Interface ResultSetMetaData	439
Interface DatabaseMetaData	440
Instructions paramétrées (PreparedStatement)	442
Extraction de données (executeQuery)	443
Mises à jour (executeUpdate)	443
Instruction LDD (execute)	444
Appels de sous-programmes	444
Appel d'une fonction	445
Appel d'une procédure	446
Transactions	447
Points de validation	447
Traitement des exceptions	449
Affichage des erreurs	449
Traitement des erreurs	450
10 Oracle et PHP	453
Configuration adoptée	453
Les logiciels	453
Les fichiers de configuration	454
Test d'Apache et de PHP	454

Test d'Apache, de PHP et d'Oracle	455
API de PHP pour Oracle (OCI)	456
Connexions	456
Constantes prédéfinies	457
Interactions avec la base	458
Extractions simples	459
Passage de paramètres	463
Traitements des erreurs	464
Procédures cataloguées	467
Métadonnées	468
API Objet PHP pour Oracle (PDO)	471
Connexions	471
Mises à jour	472
Extractions	474
Procédures cataloguées	475
11 Oracle XML DB	477
Généralités	477
Historique	477
Architecture générale	478
Répertoire logique	479
Les modes de stockage	479
Le type XMLType	480
Insertion d'un document	481
Grammaire XML Schema	483
Enregistrement de la grammaire	483
Validation totale	484
Contraintes	485
Stockage en mode object-relational	487
Annotation de la grammaire	487
Création d'une table (ou colonne) object-relational	490
Validation partielle	491
Validation totale	491
Contraintes	493
Extractions	496
La fonction XMLQuery	498
La fonction XMLCast	499
La fonction XMLTable	501
La fonction XMLExists	502
La fonction isSchemaValid	504
Mises à jour	504
Insertion d'un fragment	504
Suppression d'un fragment	505
Modification d'un fragment	506

Indexation	508
Index B-tree	509
Mode non structuré (Unstructured XMLIndex)	511
Mode structuré (Structured XMLIndex)	512
Mode mixte	513
Génération de contenus	513
Les fonctions SQL/XML	514
Conversions et analyse	517
Les fonctions d'Oracle	519
Les vues	520
Vues relationnelles	521
Vues XMLType	523
Les paquetages pour PL/SQL	526
Le paquetage DBMS_XMLGEN	527
Le paquetage DBMS_XMLSTORE	528
Le paquetage DBMS_XMLPARSER	530
Le paquetage DBMS_XMLDOM	531
XML DB Repository	532
Arborescence	533
Paquetages DBMS_XBD_REPOS	533
Les grammaires XML Schema	536
Accès par SQL	536
Les Access Control Lists (ACL)	539
Dictionnaire des données	543
12 Optimisations	545
Cadre général	545
Les acteurs	546
Contexte et objectifs	546
Présentation du jeu d'exemple	547
Les assistants d'Oracle	548
Les optimiseurs	549
L'estimateur	551
Traitement d'une instruction	552
Configuration de l'optimiseur (les hints)	552
Les statistiques destinées à l'optimiseur	553
Les histogrammes	554
Collecte	555
Outils de mesure de performances	559
Visualisation des plans d'exécution	559
L'outil tkprof	566
Paquetage DBMS_APPLICATION_INFO	571
L'utilitaire runstats de Tom Kyte	574
Bilan	576

Organisation des données	577
Des contraintes au plus près des données	577
Indexation	578
Jointures	591
Variables de lien	599
Comment réaliser des fetchs multilignes ?	601
Gestion du cache	602
Cache pour les requêtes	603
Cache pour les fonctions PL/SQL	604
Cache pour les tables	605
Tables organisées en index	607
Comparatif	607
Les débordements	608
Création d'une IOT	609
Comparaison avec une table en heap	609
Limites	610
Partitionnement	610
La clé de partition	610
Partitions par intervalle	611
Intervalles automatiques	612
Partitions par hachage	613
Partitions par liste	614
Partitions par référence	615
Sous-partitions	616
Index partitionné	617
Index partitionné local	618
Index partitionné global	619
Opérations sur les partitions et index	620
Partitionnement des tables IOT	620
Vues matérialisées	621
Réécriture de requêtes	622
Le rafraîchissement	623
Exemples	623
Dénormalisation	625
Colonnes calculées	625
Duplication de colonnes	626
Ajout de clés étrangères	627
Exemple de stratégie	627
Derniers conseils	627
Requêtes inefficaces	628
Les 10 commandements de F. Brouard	629
Index	631

Introduction

Cette introduction présente tout d'abord le cadre général dans lequel cet ouvrage se positionne (SQL, le modèle de données et l'offre d'Oracle). Vient ensuite l'utilisation des principales interfaces de commandes pour que vous puissiez programmer avec SQL dès le chapitre 1. Vous trouverez dans les compléments (sur la fiche de l'ouvrage disponible à l'adresse www.editions-eyrolles.com) les procédures d'installation de différentes versions d'Oracle pour Windows (édition *Express* ou *Enterprise*, de 9i à 12c).

SQL, une norme, un succès

C'est IBM, à tout seigneur tout honneur, qui, avec System-R, a implémenté le modèle relationnel au travers du langage SEQUEL (*Structured English as QUERY Language*) rebaptisé par la suite SQL (*Structured Query Language*).

La première norme (SQL1) date de 1987. Elle était le résultat de compromis entre constructeurs, mais fortement influencée par le dialecte d'IBM. SQL2 a été normalisée en 1992. Elle définit quatre niveaux de conformité : le niveau d'entrée (*entry level*), les niveaux intermédiaires (*transitional et intermediate levels*) et le niveau supérieur (*full level*). Les langages SQL des principaux éditeurs sont tous conformes au premier niveau et ont beaucoup de caractéristiques relevant des niveaux supérieurs. La norme SQL3 (intitulée initialement SQL:1999) comporte de nombreuses parties : concepts objets, entrepôts de données, séries temporelles, accès à des sources non SQL, réplication des données, etc. (chaque partie étant nommée ISO/IEC 9075-*i:année*, *i* allant de 1 à 14 et *année* étant la date de sortie de la dernière spécification). Une partie récente de la norme concerne la programmation côté serveur (ISO/IEC 9075-4:2011, partie 4 : *Persistent Stored Modules*).

Le succès que connaissent les grands éditeurs de SGBD relationnels (IBM, Oracle, Microsoft, Sybase et Computer Associates) a plusieurs origines et repose notamment sur SQL :

- Le langage est une norme depuis 1986 qui s'enrichit au fil du temps.
- SQL peut s'interfacer avec des langages de troisième génération comme C ou Cobol, mais aussi avec des langages plus évolués comme C++ et Java. Certains considèrent ainsi que le langage SQL n'est pas assez complet (le dialogue entre la base et l'interface n'est pas direct) et la littérature parle de « défaut d'impédance » (*impedance mismatch*).
- Les SGBD rendent indépendants programmes et données (la modification d'une structure de données n'entraîne pas forcément une importante refonte des programmes d'application).
- Ces systèmes sont bien adaptés aux grandes applications informatiques de gestion (architectures type client-serveur et Internet) et ont acquis une maturité sur le plan de la fiabilité et des performances.

- Ils intègrent des outils de développement comme les précompilateurs, les générateurs de code, d'états et de formulaires.
- Ils offrent la possibilité de stocker des informations non structurées (comme le texte, l'image, etc.) dans des champs appelés LOB (*Large Object Binary*).

Les principaux SGBD Open Source (MySQL, Firebird, Berkeley DB, PostgreSQL) ont adoptés depuis longtemps SQL pour ne pas rester en marge.

Nous étudierons les principales instructions SQL d'Oracle qui sont classifiées dans le tableau suivant.

Tableau I-1 Classification des ordres SQL

Ordres SQL	Aspect du langage
CREATE - ALTER - DROP - COMMENT - RENAME - TRUNCATE - GRANT - REVOKE	Définition des données (DDL : <i>Data Definition Language</i>).
SELECT - INSERT - UPDATE - DELETE - MERGE - LOCK TABLE	Manipulation des données (DML : <i>Data Manipulation Language</i>).
COMMIT - ROLLBACK - SAVEPOINT - SET TRANSACTION	Contrôle des transactions (TCL : <i>Transaction Control Statements</i>).

Modèle de données

Le modèle de données relationnel repose sur une théorie rigoureuse bien qu'adoptant des principes simples. La table relationnelle (*relational table*) est la structure de données de base qui contient des enregistrements, également appelés « lignes » (*rows*). Une table est composée de colonnes (*columns*) qui décrivent les enregistrements.

Tables et données

Considérons la figure suivante qui présente deux tables relationnelles permettant de stocker des compagnies, des pilotes et le fait qu'un pilote soit embauché par une compagnie :

Figure I-1 Deux tables

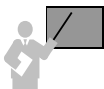
compagnie

comp	nrue	rue	ville	nom_comp
AB	1	Georges Brassens	Blagnac	Air Bus
ACTMP	24	René Lagasse	Balma	AC Toulouse

pilote

id_pil	brevet	nom_pil	nb_h_vol	compa
250	PL-1	Sarda	8500	AB
25	PL-2	Benech	5900	ACTMP
12	PL-3	Soutou	2000	ACTMP

Les clés



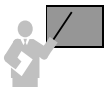
La clé primaire (*primary key*) d'une table est l'ensemble minimal de colonnes qui permet d'identifier de manière unique chaque enregistrement.

Dans la figure précédente, les colonnes « clés primaires » sont notées en gras. La colonne `comp` identifie chaque compagnie, tandis que la colonne `id_pil` permet d'identifier chaque pilote.



Une clé est dite « candidate » (*candidate key*) si elle peut se substituer à la clé primaire à tout instant. Une table peut contenir plusieurs clés candidates ou aucune.

Dans l'exemple, la colonne `brevet` pourrait jouer le rôle d'une clé candidate, car il est probable que chaque numéro de brevet soit unique. Pour les compagnies, le nom (`nom_comp`) s'il est supposé unique peut également jouer le rôle de clé candidate.



Une clé étrangère (*foreign key*) référence dans la majorité des cas une clé primaire d'une autre table (sinon une clé candidate sur laquelle un index unique aura été défini). Une clé étrangère est composée d'une ou plusieurs colonnes. Une table peut contenir plusieurs clés étrangères ou aucune.

La colonne `compa` (notée en italique dans la figure) est une clé étrangère, car elle permet de référencer un enregistrement unique de la table `compagnie` via la clé primaire `comp`.

Le modèle relationnel est ainsi fondamentalement basé sur les valeurs. Les associations entre tables sont toujours binaires et assurées par les clés étrangères. Les théoriciens considèrent celles-ci comme des pointeurs logiques. Les clés primaires et étrangères seront définies dans les tables en SQL à l'aide de contraintes.

Oracle

Il sera très difficile, pour ne pas dire impossible, à un autre éditeur de logiciels de trouver un nom mieux adapté à la gestion des données que celui d'« Oracle ». Ce nom semble prédestiné à cet usage ; citons *Le Petit Larousse* :

ORACLE *n.m.* (*lat. oraculum*) **ANTIQ.** Réponse d'une divinité au fidèle qui la consultait ; divinité qui rendait cette réponse ; sanctuaire où cette réponse était rendue. **LITT.** Décision jugée infaillible et émanant d'une personne de grande autorité ; personne considérée comme infaillible.

Oracle représenterait ainsi à la fois une réponse infaillible, un lieu où serait rendue cette réponse et une divinité. Rien que ça ! Tout cela peut être en partie vérifié si votre conception

est bien faite, vos données insérées cohérentes, vos requêtes et programmes bien écrits. Ajoutons aussi le fait que les ordinateurs fonctionnent bien et qu'une personne compétente se trouve au support. C'est tout le mal que nous vous souhaitons.

Oracle Corporation, société américaine située en Californie, développe et commercialise un SGBD et un ensemble de produits de développement. Oracle a des filiales dans un grand nombre de pays. Initialement composée de cinq départements (marketing, commercial, avant-vente, conseil et formation), la filiale française (Oracle France) a été créée en 1986. Le département formation a été dissous en 2010, donnant naissance à la société EASYTEAM (premier partenaire Platinum en France), composée des ex-formateurs d'Oracle France.

Un peu d'histoire

En 1977, Larry Ellison, Bob Miner et Ed Oates fondent la société *Software Development Laboratories* (SDL). L'article de Edgar Frank Codd (1923-2003), « A Relational Model of Data for Large Shared Data Banks », *Communications of the ACM* paru en 1970, fait devenir le mathématicien et ancien pilote de la RAF durant la Seconde Guerre mondiale, inventeur du modèle relationnel et de SQL. Les associés de SDL devinent le potentiel des concepts de Codd et se lancent dans l'aventure en baptisant leur logiciel « Oracle ». En 1979, SDL devient *Relational Software Inc.* (RSI) qui donnera naissance à la société *Oracle Corp.* en 1983. La première version du SGBD s'appelle RSI-1 et utilise SQL. Le tableau suivant résume la chronologie des versions.

Tableau I-2 Chronologie des versions d'Oracle

1979 Oracle 2	Première version commerciale écrite en C/assembleur pour Digital – pas de mode transactionnel.
1983 Oracle 3	Réécrit en C – verrous.
1984 Oracle 4	Portage sur IBM/VM, MVS, PC – transaction (lecture consistante).
1986 Oracle 5	Architecture client-serveur avec SQL*Net – version pour Apple.
1988 Oracle 6	Verrouillage niveau ligne – sauvegarde/restauration – AGL – PL/SQL.
1991 Oracle 6.1	<i>Parallel Server</i> sur DEC.
1992 Oracle 7	Contraintes référentielles – procédures cataloguées – déclencheurs – version Windows en 1995.
1994	Serveur de données vidéo.
1995	Connexions sur le Web.
1997 Oracle 8	Objet-relationnel – partitionnement – LOB – Java.
1998 Oracle8 <i>i</i>	<i>i</i> comme Internet, SQLJ – Linux – XML.
2001 Oracle9 <i>i</i>	Services Web – serveur d'applications – architectures sans fil.
2004 Oracle 10 <i>g</i>	<i>g</i> comme <i>Grid computing</i> (ressources en <i>clusters</i>).
2007 Oracle 11 <i>g</i>	Auto-configuration.
2013 Oracle 12 <i>c</i>	Architecture multitenant, Cloud et Big Data.

Avec IBM, Oracle a fait un pas vers l'objet en 1997, mais cette approche ne compte toujours pas parmi les priorités des clients d'Oracle. L'éditeur met plus en avant ses aspects transactionnels, décisionnels, de partitionnement et de réplication. Les technologies liées à Java, bien qu'elles soient largement présentes sous Oracle9i, ne constituent pas non plus la majeure partie des applicatifs exploités par les clients d'Oracle.

La version 10g renforce le partage et la coordination des serveurs en équilibrant les charges afin de mettre à disposition des ressources réparties (répond au concept de l'informatique à la demande). Cette idée est déjà connue sous le nom de « mise en grappe » des serveurs (*clustering*). Une partie des fonctions majeures de la version 10g est présente dans la version 9i RAC (*Real Application Cluster*).

La version 11g Oracle insiste sur les capacités d'auto-diagnostic, d'auto-administration et d'auto-configuration pour optimiser la gestion de la mémoire et pour pouvoir faire remonter des alertes de dysfonctionnement. En raison des exigences en matière de traçabilité et du désir de capacité de décision (*datamining*), la quantité de données gérées par les SGBD triplant tous les deux ans, 11g met aussi l'accent sur la capacité à optimiser le stockage.

La version 12c bouleverse l'architecture d'une instance assimilée à une base unique en introduisant l'architecture multitenant capable d'héberger plusieurs bases de données enfichables (*pluggable database*) dans une base de données de conteneur multipropriétaire (*container database*).

Rachat de Sun (et de MySQL)

Contrairement à la rumeur du début de 2007, MySQL n'entre pas en Bourse, il est racheté pour un milliard de dollars en janvier 2008 par Sun Microsystems déjà propriétaire de Java. Sun arrive ainsi sur un segment où il était absent, aux côtés d'Oracle, d'IBM et de Microsoft.

Craignant l'achat de Sun par IBM et redoutant HP dans le haut de gamme Unix, Oracle se repositionne dans le hardware et sur le marché des services pour *datacenters* en avril 2009, en achetant Sun. Ce sont aussi les langages Java et le système d'exploitation Solaris qui ont pesé dans la balance. En effet, c'est sur Solaris, et non sur Linux, que sont déployés le plus grand nombre de serveurs Oracle.

Il faudra attendre novembre 2009 pour que la Commission européenne confirme son refus de la fusion entre Oracle et Sun, suspectant que le rachat de MySQL aboutisse à une situation de quasi monopole sur le marché des SGDB. En décembre 2009, avec le soutien de quelque 59 sénateurs américains, Oracle publie 10 engagements concernant toutes les zones géographiques et pour une durée de 5 ans.

1. Assurer aux utilisateurs le choix de leur moteur (*MySQL's Pluggable Storage Engine Architecture*).
2. Ne pas changer les clauses d'utilisation d'une manière préjudiciable à un fournisseur tiers.
3. Poursuivre les accords commerciaux contractés par Sun.

4. Garder MySQL sous licence GPL.
5. Ne pas imposer un support des services d'Oracle aux clients du SGBD.
6. Augmenter les ressources allouées à la R&D de MySQL.
7. Créer un comité d'utilisateurs pour, dans les 6 mois, étudier les retours et priorités de développement de MySQL.
8. Créer ce même comité pour les fournisseurs de solutions incluant MySQL.
9. Continuer d'éditer, mettre à jour et distribuer gratuitement le manuel d'utilisation du SGBD.
10. Laisser aux utilisateurs le choix de la société qui assurera le support de MySQL.

Considérant d'une part ces engagements, et d'autre part l'existence de concurrents (notamment IBM, Microsoft et PostgreSQL dans le monde du libre), la Commission européenne avale la fusion fin janvier 2010 pour un montant de 7,4 milliards de dollars. Cinq ans après (en 2015), MySQL est toujours dans le giron d'Oracle et, selon un dirigeant d'Oracle, les effectifs dédiés au développement et au support de MySQL ont doublé en cinq ans, et ceux de l'assurance qualité ont triplé.

Offre du moment





La page d'accueil d'Oracle (www.oracle.com) focalise sur les technologies à la mode (pour le moment le cloud). Sans parler des matériels, services, support, progiciels, etc., la base de données semble n'être qu'une brique à l'offre tentaculaire...

Figure I-2 Offre Oracle

Oracle Cloud Oracle Mobile Applications Customer Experience Enterprise Performance Management Enterprise Resource Planning Human Capital Management Supply Chain Management Industry Applications Applications Product Lines Database Oracle Database Oracle Database In-Memory Oracle Multitenant Real Application Clusters Data Warehousing Database High Availability Database Security MySQL Oracle NoSQL Database TimesTen In-Memory Database Java	Operating Systems Oracle Solaris Oracle Linux Business Analytics Middleware Cloud Application Foundation Data Integration Identity Management Mobile Platform Service-Oriented Architecture Business Process Management WebCenter WebLogic Enterprise Management Cloud Management Application Management Database Management Middleware Management Hardware and Virtualization Management Heterogeneous Management Lifecycle Management	Engineered Systems Big Data Appliance Exadata Database Machine Exalogic Elastic Cloud Exalytics In-Memory Machine Database Appliance Oracle SuperCluster Oracle Virtual Compute Appliance Oracle ZFS Storage Appliance Servers SPARC x86 Blade Netra Storage and Tape SAN Storage NAS Storage Tape Storage Networking and Data Center Fabric Products Enterprise Communications	Virtualization Oracle Secure Global Desktop Oracle VM Server for x86 Oracle VM Server for SPARC Services Consulting Premier Support Advanced Customer Support Training Cloud Services Financing Oracle Customer Programs Customer and Partner Successes Products A-Z List Oracle Products from Acquired Companies Product Price List
---	---	---	--

Depuis la version 10g, les principales éditions du produit Oracle Database ont pour nom *Entreprise*, *Standard* et *Standard One*. Le produit monoposte est qualifié de *Personal* et la version gratuite de *Express*.

Figure I-3 Éditions d'Oracle Database ; extrait du site

				
	Oracle Database Express Edition Download Now	Oracle Database Standard Edition One Price Now	Oracle Database Standard Edition Price Now	Oracle Database Enterprise Edition Price Now
Maximum	1 CPU	2 Sockets	4 Sockets	No Limit
RAM	1GB	OS Max	OS Max	OS Max
Database Size	11GB	No Limit	No Limit	No Limit
Oracle Multitenant				Option

Un grand nombre d'options (payantes en sus de la base et en fonction de l'édition) permettent de renforcer, notamment, les performances, la sécurité, le traitement transactionnel et le *datawarehouse*. Citons les plus connues : *Data Guard*, *Real Application Clusters*, *Partitioning*, *Advanced Security*, *Advanced Compression*, *Diagnostics Pack*, *Tuning Pack*, *OLAP*, *Data Mining* et *Spatial*.

Les prix des licences, clairement affichés sur le site d'Oracle, permettent deux modes de calcul : en fonction du nombre d'utilisateurs nommés (*named user plus*) ou du nombre de processeurs (*processors*). Le premier calcul convient généralement à des applications en mode client-serveur, le second serait davantage adapté aux architectures multitiens et Web.

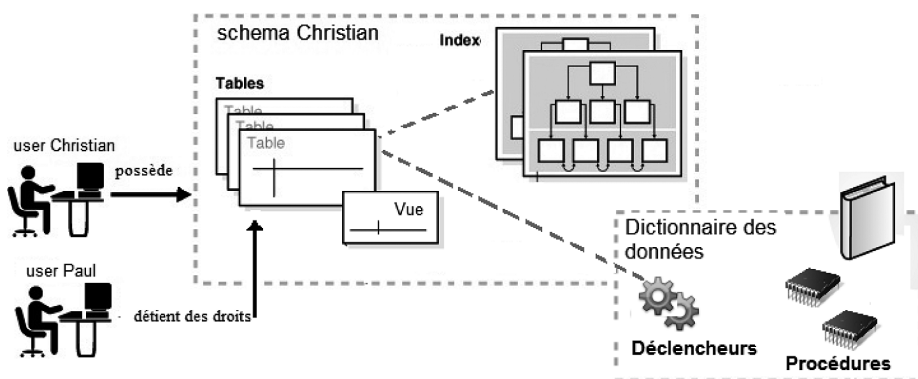
Figure I-4 Prix d'Oracle 2015 ; extrait du site

Database Products	Oracle Database				Prices in USA (Dollar)
	Named User Plus	Software Update License & Support	Processor License	Software Update License & Support	
Oracle Database					
Standard Edition One	180	39.60	5,800	1,276.00	
Standard Edition	350	77.00	17,500	3,850.00	
Enterprise Edition	950	209.00	47,500	10,450.00	
Personal Edition	460	101.20	-	-	
Mobile Server	-	-	23,000	5,060.00	
NoSQL Database Enterprise Edition	200	44	10,000	2,200.00	
Enterprise Edition Options:					
Multitenant	350	77.00	17,500	3,850.00	
Real Application Clusters	460	101.20	23,000	5,060.00	
Real Application Clusters One Node	200	44.00	10,000	2,200.00	
Active Data Guard	230	50.60	11,500	2,530.00	
Partitioning	230	50.60	11,500	2,530.00	
Real Application Testing	230	50.60	11,500	2,530.00	
Advanced Compression	230	50.60	11,500	2,530.00	
Advanced Security	300	66.00	15,000	3,300.00	
Label Security	230	50.60	11,500	2,530.00	

Notion de schéma

Au niveau d'une base de données, qu'elle soit conventionnelle, enfichable ou conteneur (avec l'option multitenant de la version 12c), un schéma ne se distingue d'un utilisateur (*user*) que parce qu'il contient des objets (table, index, vue, etc.). Ainsi, chaque objet d'une base est associé à son propriétaire (l'utilisateur qui l'a créé ; c'est le cas de l'utilisateur Christian de la figure suivante). S'il ne détient aucun objet, un user peut être perçu simplement comme un identificateur de connexion (c'est le cas de Paul). Tout utilisateur peut toutefois accéder à des objets ne lui appartenant pas, sous réserve d'avoir reçu des droits accordés par le propriétaire ou un administrateur.

Figure I-5 Schéma et utilisateur



Tous les éléments d'un schéma ne seront pas étudiés car certains sont très spécifiques et sortent du cadre traditionnel de l'apprentissage. Le tableau suivant indique dans quel chapitre du livre vous trouverez les principaux éléments d'un schéma :

Tableau I-3 Éléments d'un schéma Oracle

Éléments étudiés – Chapitre	Aspects non étudiés
Déclencheurs (<i>triggers</i>) – 7	Dimensions (cubes)
Fonctions et procédures cataloguées, paquetages – 7	Liens de bases de données (<i>database links</i>)
Librairies de procédures externes – site d'accompagnement	Opérateurs
Index – 1, 12	Tables, types et vues objets
Java – 9, site d'accompagnement	Spatial
Séquences et synonymes – 2, 5	
Tables et tables en index – 1	
Vues (<i>views</i>) – 5	
XML – 11	
Clusters – 12	
Partitions – 12	
Vues matérialisées – 12	

Accès à Oracle depuis Windows

Après avoir installé Oracle sur votre ordinateur, vous serez libre de choisir l'accès qui vous convient (le client SQL comme on dit). Ce livre utilise principalement l'interface SQL*Plus (livrée avec la base et dans toutes les versions clientes d'Oracle). Vous pouvez opter pour SQL Developer (produit Java gratuit sur le site d'Oracle qui ne nécessite aucune installation, simplement une décompression dans un de vos répertoires), ou pour un programme Java (via JDBC) ou PHP.

Il existe d'autres clients SQL qui sont payants ou gratuits ; citons SQL Developer, SQLTools, SQL Navigator et TOAD le plus renommé et probablement le plus performant.

Détail d'un numéro de version

Détaillons la signification du numéro de la version 11.1.0.6.0 (première *release* de la 11g disponible sous Windows) :

- 11 désigne le numéro de la version majeure de la base ;
- 1 désigne le numéro de version de la maintenance ;
- 0 désigne le numéro de version du serveur d'applications ;
- 6 désigne le numéro de version du composant (*patch*) ;
- 0 est le numéro de la version de la plate-forme.

Vous pourrez contrôler la version de l'interface SQL*Plus et celle du serveur à l'issue de votre première connexion comme le montre la figure I-6 (ici, les versions de l'outil client et du serveur sont identiques car l'installation du serveur inclut l'installation de l'interface de même version).

Figure I-6 Version du serveur et du client

```

SQL Plus
SQL*Plus: Release 12.1.0.1.0 Production on Jeu. Nov. 6 05:57:36 2014
Copyright (c) 1982, 2013, Oracle. All rights reserved.

Entrez le nom utilisateur : system
Entrez le mot de passe :
Heure de la dernière connexion réussie : Jeu. Nov. 06 2014 05:51:16 +01:00

Connect@ >
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options

SQL>

```

Les clients SQL

Les clients SQL permettent de dialoguer avec la base de différentes manières :

- exécution de commandes SQL, SQL*Plus et de blocs PL/SQL ;
- échanges de messages avec d'autres utilisateurs ;
- création de rapports d'impression en incluant des calculs ;
- réalisation des tâches d'administration en ligne.

SQL*Plus

En fonction de la version d'Oracle dont vous disposez, plusieurs interfaces SQL*Plus peuvent être disponibles sous Windows :

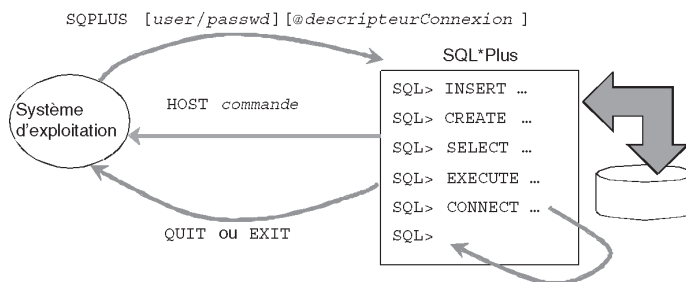
- en mode ligne de commande (qui ressemble à une fenêtre DOS ou telnet) ;
- avec l'interface graphique (qui est la plus proche du monde Windows) ;
- avec l'interface graphique SQL*Plus Worksheet de l'outil *Enterprise Manager* (plus évoluée que la précédente) ;
- avec le navigateur via l'interface web *iSQL*Plus* (*i* comme « Internet » ; cette interface s'apparente assez à celle de EasyPHP en étant très intuitive).



Du fait que les interfaces graphiques et web aient été abandonnées depuis la version 11g, utilisez toujours l'interface en ligne de commande qui restera nécessairement disponible pour les versions à venir.

Le principe général de ces interfaces est le suivant : après une connexion locale ou distante, des instructions sont saisies et envoyées à la base qui retourne des résultats affichés dans la même fenêtre de commandes. La commande SQL*Plus `HOST` permet d'exécuter une commande du système d'exploitation qui héberge le client Oracle (exemple : `DIR` sous Windows ou `ls` sous Unix).

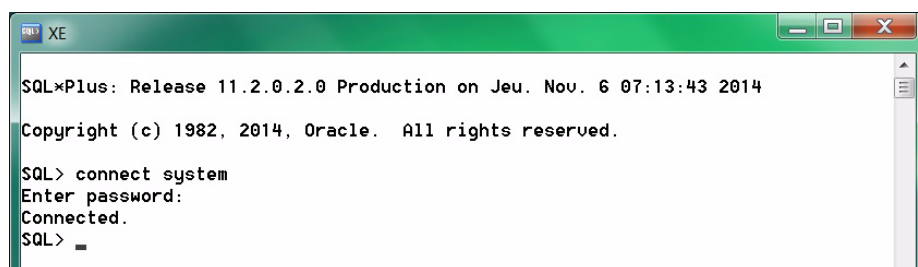
Figure I-7 Principe général des interfaces SQL*Plus



Connexion à Oracle

Quel que soit le mode de connexion que vous allez choisir, vous devrez toujours renseigner le nom de l'utilisateur et son mot de passe. D'autres paramètres peuvent être nécessaires comme le nom ou l'adresse du serveur, un numéro de port, un nom d'instance ou de service. Commençons par faire simple et utilisons l'interface SQL*Plus pour connecter l'utilisateur *system* à l'aide du mot de passe donné lors de l'installation, vous devez obtenir un résultat analogue (si vous disposez d'une version *Express* ; sinon, vous visualiserez davantage d'informations concernant la version du serveur).

Figure I-8 Connexion à Oracle



```
SQL*Plus: Release 11.2.0.2.0 Production on Jeu. Nov. 6 07:13:43 2014
Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> connect system
Enter password:
Connected.
SQL> _
```

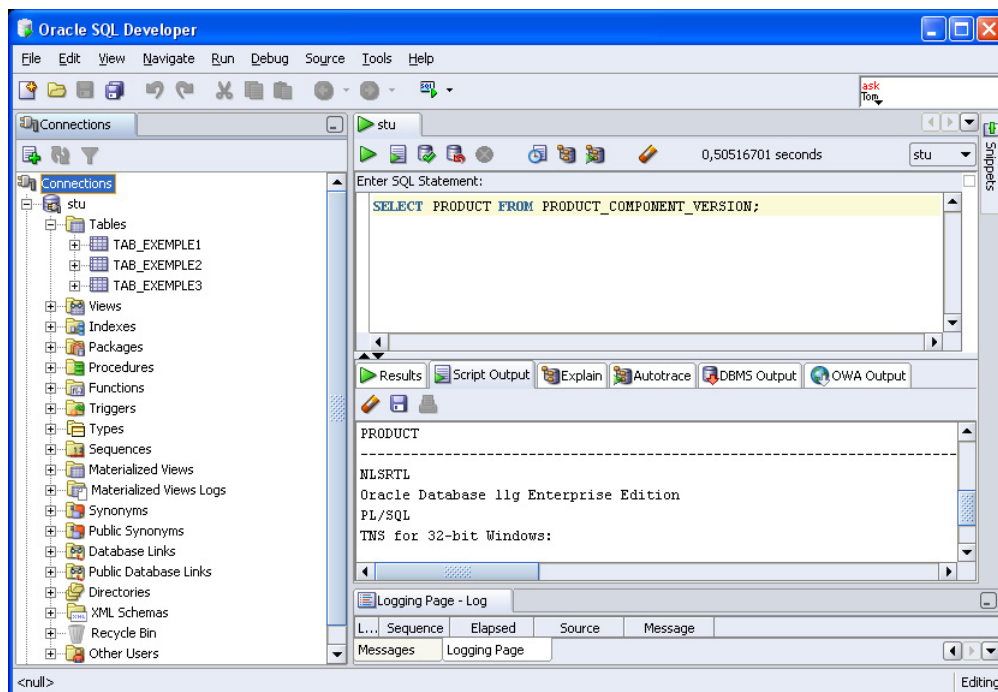
SQL Developer

SQL Developer est un outil gratuit de développement (écrit en Java) disponible sur le site d'Oracle (www.oracle.com/technetwork/developer-tools/sql-developer). Différentes versions sont disponibles (Windows 32 et 64 bits, Mac et Linux RPM et autres)...

Depuis la version 3, cet outil inclut un générateur de requêtes (*query builder*) et un gestionnaire de jobs (*schedule builder*). De plus, il est possible d'analyser sommairement des performances de requêtes (*Explain*, *Autotrace* et *SQL Tuning Advisor*). Des assistants d'exportation (par exemple, au format CSV) et d'importation (par exemple, de données Excel) sont également disponibles. Il permet même de visualiser et de manipuler des données spatiales et décisionnelles (*Spatial* et *Data Miner*).

Depuis la première version 4, une vue DBA permet d'administrer en partie une base (paramètres de configuration, backup et recovery avec RMAN, exportation et importation, comptes utilisateur, profils, rôles, privilèges, etc.). Utilisé conjointement avec le pack Tuning et Diagnostic, il est possible de visualiser l'activité (avec ADDM, AWR et ASH).

Figure I-9 SQL Developer



Après avoir téléchargé SQL Developer, vous n'aurez qu'à décompresser l'archive dans le répertoire Programmes et à exécuter `sqldeveloper.exe`. Mises à part les éditions *Express* d'Oracle, SQL Developer est inclus dans les éditions *Standard* et *Enterprise*, et se trouve dans le menu Démarrer Oracle.../Développement d'applications. À la première exécution, le chemin du JDK vous sera probablement demandé.



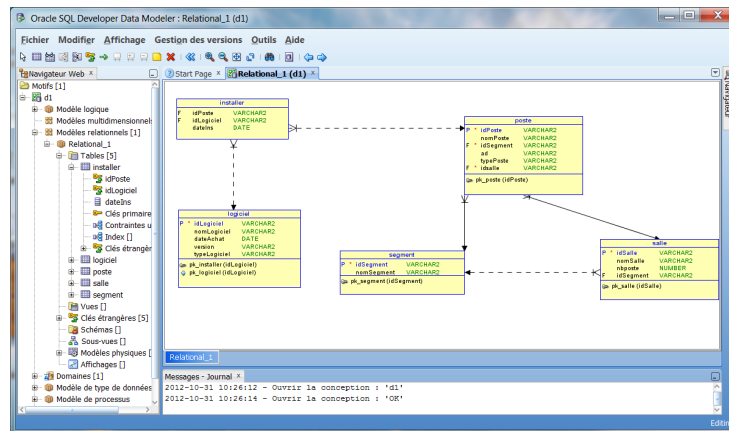
Bien que l'outil permette un grand nombre de fonctionnalités, certaines commandes SQL*Plus (GET, START, COL, ACCEPT...) ne sont pas opérantes.

SQL Data Modeler

Un outil de développement SQL n'est pas forcément un outil de conception. Ce dernier vise à construire ou « cartographier » des tables alors que le premier les manipule. Les outils de conception sont nombreux mais le plus souvent payants (TOAD Data Modeler, PowerAMC, DeZign for Databases, ERwin Data Modeler, Enterprise Architect, ER/Studio, Navicat, etc.). Oracle fournit gratuitement SQL Data Modeler (www.oracle.com/technetwork/developer-tools/datamodeler) qui est construit sur une interface analogue à son homologue SQL Developer. Le

niveau conceptuel des données n'est pas le plus abouti mais si vous travaillez uniquement au niveau des tables, contraintes et index, il vous conviendra sans doute. Il vous permettra de générer des scripts de création de tables ou de visualiser graphiquement des tables d'un schéma, ce qui est très intéressant pour la compréhension et pour écrire des requêtes réalisant des jointures cohérentes (voir le chapitre 4).

Figure I-10 SQL Developer Data Modeler



Premiers pas



Débutez votre apprentissage avec l'interface SQL*Plus afin de vous familiariser avec les manipulations basiques qui vous serviront fréquemment par la suite, car il y a de grandes chances pour que cette interface soit présente dans les différents environnements que vous fréquenterez. Si vous commencez par *SQL Developer*, vous n'utiliserez plus SQL*Plus et le jour où vous ne disposerez que de cette interface, vous risquez d'être bloqué et de ne pas pouvoir fournir les résultats attendus...

Création d'un utilisateur

Pour créer un utilisateur, utilisez le script `CreaUtilisateur.sql` situé dans le répertoire `Introduction`. Choisissez-lui ensuite un nom (supprimer les caractères < et >) ainsi qu'un mot de passe. Si vous enregistrez ce fichier avec un autre nom dans un autre répertoire, il est préférable de ne pas utiliser de caractères spéciaux (ni d'espaces) dans le nom de vos répertoires.

Une fois connecté, exécutez votre script dans l'interface SQL*Plus grâce à la commande `start chemin/nom_script` (par exemple, `start C:\temp\cre_eyrolles.sql`). L'écran suivant concerne la création d'un utilisateur dans la base enclenchable (PDBORCL par

défaut). Pour des éditions antérieures à la version 12c, les trois instructions encadrées ne sont pas à exécuter.

Figure I-11 Création d'un utilisateur

```

SQL*Plus: Release 12.1.0.1.0 Production on Jeu. Nov. 6 11:53:01 2014
Copyright (c) 1982, 2013, Oracle. All rights reserved.
Entrez le nom utilisateur : system
Entrez le mot de passe :
Connecté à :
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing

SQL> conn sys/ *** ← AS sysdba mot de passe de l'installation
Connecté.

SQL> alter pluggable database PDBORCL open;
Base de données pluggable modifiée.

SQL> alter session set container=PDBORCL;
Session modifiée.

SQL> start C:\temp\cre_eyrolles.sql

```

Vous devez obtenir les deux messages suivants (aux accents près) :

```

Utilisateur créé.
Autorisation de privilèges (GRANT) acceptée.

```

Voilà, votre utilisateur est créé, il peut se connecter et possède les prérogatives minimales pour exécuter la plupart des commandes décrites dans cet ouvrage.



Si vous voulez afficher vos instructions avant qu'elles ne s'exécutent sous SQL*Plus (utile pour tracer l'exécution de plusieurs commandes), lancez la commande `set echo on` qui restera valable pour toute la session.

À l'instar de la syntaxe du langage SQL d'Oracle, les commandes SQL*Plus sont insensibles à la casse. Dans cet ouvrage, elles sont en général mentionnées en majuscules.

Commandes basiques de SQL*Plus

Le tableau I-4 récapitule les commandes qui permettent de manipuler le buffer de l'interface SQL*Plus. Une fois écrite dans l'interface, la commande (qui peut constituer une instruction SQL ou un bloc PL/SQL) pourra être manipulée, avant ou après son exécution.

Tableau I-4 Commandes du buffer d'entrée (pas pour /SQL*Plus)

Commande	Commentaires
R	Exécute (<i>run</i>).
L	Liste le contenu du buffer.
L*	Liste la ligne courante.
L <i>n</i>	Liste la <i>nième</i> ligne du buffer qui devient la ligne courante.
I	Insère une ligne après la ligne courante.
A <i>texte</i>	Ajoute <i>texte</i> à la fin de la ligne courante.
DEL	Supprime la ligne courante.
C/ <i>texte1/texte2</i> /	Substitution de la première occurrence de <i>texte1</i> par <i>texte2</i> dans la ligne courante.
CLEAR	Efface le contenu du buffer.
QUIT ou EXIT	Quitte SQL*Plus.
CONNECT <i>user</i> / <i>password@descripteur</i>	Autre connexion (sans sortir de l'interface).
GET <i>fichier</i>	Charge dans le buffer le contenu du <i>fichier.sql</i> qui se trouve dans le répertoire courant.
SAVE <i>fichier</i>	Écrit le contenu du buffer dans <i>fichier.sql</i> qui se trouve dans le répertoire courant.
START <i>fichier</i> ou @ <i>fichier</i>	Charge dans le buffer et exécute <i>fichier.sql</i> .
SPOOL <i>fichier</i>	Crée <i>fichier.lst</i> dans le répertoire courant qui va contenir la trace des entrées/sorties jusqu'à la commande SPOOL OFF.

Variables d'environnement

Les variables d'environnement (voir le tableau I-5) vous permettront de paramétrer votre session SQL*Plus. L'affectation d'une variable s'opère avec SET (ou par un menu si vous utilisez encore l'interface Windows graphique). À tout moment, la commande SHOW *nom_variable* vous renseignera à propos d'une variable d'environnement (voir le tableau I-6).

Tableau I-5 Variables d'environnement

Commande	Commentaires
SET AUTOCOMMIT {ON OFF IMMEDIATE n}	Validation automatique après une ou <i>n</i> commandes.
SET ECHO {ON OFF}	Affichage des commandes avant exécution.
SET LINESIZE {80 n}	Taille en caractères d'une ligne de résultats.
SET PAGESIZE {24 n}	Taille en lignes d'une page de résultats.
SET SERVEROUT [PUT] {ON OFF}	Activation de l'affichage pour tracer des exécutions.
SET TERMOUT {ON OFF}	Affichage des résultats.
SET TIME {ON OFF}	Affichage de l'heure dans le prompt.

Tableau I-6 Paramètres de la commande SHOW

Paramètre	Commentaires
<i>variableEnvironnement</i>	Variable d'environnement (AUTOCOMMIT, ECHO, etc.).
ALL	Toutes les variables d'environnement.
ERRORS	Erreurs de compilation d'un bloc ou d'un sous-programme.
RELEASE	Version du SGBD utilisé.
USER	Nom de l'utilisateur connecté.

À propos des accents et jeux de caractères

Il est possible de paramétrer sous Oracle certains formats, tels que la date, l'heure, les jours de la semaine, la monnaie, le jeu de caractères, etc. La principale difficulté étant que ces paramètres peuvent être différents entre le serveur Oracle, les systèmes d'exploitation hébergeant la base et le programme client (client Oracle natif comme l'interface SQL*Plus ou client utilisant un pilote Oracle JDBC par exemple).

En soit cette différence n'est pas dangereuse car Oracle opère les conversions automatiquement, mais il est important de savoir quel format de données le client attend. Une base de données peut stocker des prix en dollars car son jeu de caractères est américain et les restituer en euros car le client est européen. Bien sûr le chiffre stocké en base est en valeur d'euros et s'il est affiché par un client local il apparaîtra sous la forme de dollar. Ce raisonnement vaut pour les dates et accents.

Configuration côté serveur

Pour connaître la configuration côté serveur (instance sur laquelle vous êtes connecté), il faut interroger la vue `NLS_DATABASE_PARAMETERS` qui renseigne, entre autres, la langue, au territoire (pour le format des dates, des monnaies) et au jeu de caractères. Dans cet exemple, la base installée est Oracle 10g *Express Edition*.

```
SQL> SELECT PARAMETER, VALUE FROM NLS_DATABASE_PARAMETERS
        WHERE parameter IN ('NLS_LANGUAGE', 'NLS_TERRITORY',
        'NLS_CHARACTERSET', 'NLS_CURRENCY');
```

PARAMETER	VALUE
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
NLS_CURRENCY	\$
NLS_CHARACTERSET	AL32UTF8

Il apparaît la langue anglaise (AMERICAN) de l'instance, les codes américains pour le format des dates, des monnaies (AMERICA) et le jeu de caractères par défaut (AL32UTF8) qui est une extension (pour les plates-formes ASCII) du classique Unicode UTF-8 codé sur 4 octets.

Configuration côté client

Pour connaître la configuration côté client (ici une session SQL*Plus), il faut interroger la vue `NLS_SESSION_PARAMETERS` qui renseigne un certain nombre de paramètres mais pas le jeu de caractères.

```
SQL> SELECT PARAMETER, VALUE FROM NLS_SESSION_PARAMETERS
        WHERE parameter IN
        ('NLS_LANGUAGE', 'NLS_TERRITORY', 'NLS_CURRENCY');
```

PARAMETER	VALUE
NLS_LANGUAGE	FRENCH
NLS_TERRITORY	FRANCE
NLS_CURRENCY	Ç

Il apparaît que le client a été installé en choisissant la langue française avec ses conventions (notamment pour le format de dates et de la monnaie). Le jeu de caractères n'est pas ici accessible. Le symbole € n'est pas restitué car certaines interfaces SQL*Plus utilisent une police de caractère de type Courier qui n'inclut pas ce symbole.

Afin de pouvoir restituer des caractères accentués :

- Concernant le client SQL*Plus en mode ligne de commande, il faut affecter la variable NLS_LANG (sous Windows `set NLS_LANG=FRENCH_FRANCE.WE8PC850`, par exemple avec Unix `export NLS_LANG=...`).
- Pour les autres clients graphiques Windows tels que *SQL Developer*, vous devrez vous assurer que la base de registres contient la valeur FRENCH_FRANCE.WE8MSWIN1252 pour la clé NLS_LANG (choix Edition/Rechercher... en lançant regedit).

Une fois ceci fait, vous devriez pouvoir gérer les accents au niveau des données, tables, colonnes, etc. L'exemple suivant illustre cette possibilité (ici le test est réalisé dans la console *SQL Developer*).

Figure I-12 Restitution de caractères accentués

```

Enter SQL Statement:
CREATE TABLE tableAccentuée (nom VARCHAR2(15), établissement VARCHAR(19),
                             prime NUMBER, devise VARCHAR2(6));
INSERT INTO tableAccentuée VALUES ('éric Moreau', 'Collège J. Jaurès', 123.45, '€');
INSERT INTO tableAccentuée VALUES ('agnès Bidal', 'Lycée à Pau', 145.6, '$');
SELECT nom,établissement,prime,devise FROM tableAccentuée ;

Results | Script Output | Explain | Autotrace | DBMS Output | OWA Output
-----|-----|-----|-----|-----|-----
CREATE TABLE succeeded.
1 rows inserted
1 rows inserted
NOM          ÉTABLISSEMENT      PRIME      DEVISE
-----|-----|-----|-----
éric Moreau   Collège J. Jaurès  123,45     €
agnès Bidal   Lycée à Pau        145,6      $
  
```

Partie I

SQL de base

Chapitre 1

Définition des données

Ce chapitre décrit les instructions SQL qui constituent l'aspect LDD (langage de définition des données) de SQL. À cet effet, nous verrons notamment comment déclarer une table, ses éventuels contraintes et index.

Tables relationnelles

Une table est créée en SQL par l'instruction `CREATE TABLE`, modifiée au niveau de sa structure par l'instruction `ALTER TABLE` et supprimée par la commande `DROP TABLE`.

Création d'une table (CREATE TABLE)

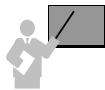
Pour pouvoir créer une table dans votre schéma, il faut que vous ayez reçu le privilège `CREATE TABLE`. Si vous avez le privilège `CREATE ANY TABLE`, vous pouvez créer des tables dans tout schéma. Le mécanisme des privilèges est décrit au chapitre « Contrôle des données ».

La syntaxe SQL simplifiée est la suivante :

```
CREATE TABLE [schéma.]nomTable
  (colonne1 type1 [DEFAULT valeur1] [NOT NULL]
  [, colonne2 type2 [DEFAULT valeur2] [NOT NULL] ]
  [CONSTRAINT nomContrainte1 typeContrainte1]... ) ;
```

- *schéma* : s'il est omis, il sera assimilé au nom de l'utilisateur connecté. S'il est précisé, il désigne soit l'utilisateur courant soit un autre utilisateur de la base (dans ce cas, il faut que l'utilisateur courant ait le droit de créer une table dans un autre schéma). Nous aborderons ces points dans le chapitre 5 et nous considérerons jusque-là que nous travaillons dans le schéma de l'utilisateur couramment connecté (ce sera votre configuration la plupart du temps).
- *nomTable* : peut comporter au maximum 30 caractères (lettres, chiffres et caractères `_`, `$` et `#`). Si l'identificateur n'est pas encadré par des guillemets (*quoted identifier*), le nom est insensible à la casse et sera converti en majuscules dans le dictionnaire de données (il en va de même pour le nom des colonnes).

- *colonnei typei* : nom de colonne et son type (NUMBER, VARCHAR2, DATE...). L'option DEFAULT fixe une valeur en cas de non-renseignement (NULL). L'option NOT NULL interdit que la valeur de la colonne ne soit pas renseignée.



Le marqueur NULL ne désigne pas une valeur mais une absence de valeur qu'on peut traduire comme non disponible, non affectée, inconnue ou inapplicable. NULL est différent d'une chaîne vide, d'un zéro ou des espaces. Ce marqueur est à étudier de près car, dans bien des cas, deux NULL ne sont pas identiques. Les requêtes d'extraction peuvent renvoyer des résultats aberrants si les NULL sont mal interprétés. En positionnant le plus possible de NOT NULL dans vos colonnes, vous diminuerez les traitements additionnels à opérer par la suite.

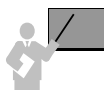
- *nomContrainte* *typeContrainte* : noms de la contrainte et son type (clé primaire, clé étrangère, etc.). Nous allons détailler dans le paragraphe suivant les différentes contraintes possibles.
- ; : symbole qui termine une instruction SQL d'Oracle. Le slash (/) peut également terminer une instruction à condition de le placer à la première colonne de la dernière ligne.

Casse et commentaires

Dans toute instruction SQL (déclaration, manipulation, interrogation et contrôle des données), il est possible d'inclure des retours chariots, des tabulations, espaces et commentaires (sur une ligne précédée de deux tirets --, sur plusieurs lignes entre /* et */). De même, la casse n'a pas d'importance au niveau des mots-clés de SQL, des noms de tables, colonnes, index, etc. Les scripts suivants décrivent la déclaration d'une même table en utilisant différentes conventions :

Tableau 1-1 Différentes écritures SQL

Sans commentaire	Avec commentaires
<pre>CREATE TABLE MêmesévènementsàNoël (colonne CHAR);</pre>	<pre>CREATE TABLE -- nom de la table TEST(-- description COLONNE NUMBER(38,8)) -- fin, ne pas oublier le point-virgule. ;</pre>
<pre>CREATE TABLE Test (colonne NUMBER(38,8));</pre>	<pre>CREATE TABLE Test (/* une plus grande description des colonnes */ COLONNE NUMBER(38,8));</pre>
<pre>CREATE table test (Colonne NUMBER(38,8));</pre>	



La casse a une incidence majeure dans les expressions de comparaison entre colonnes et valeurs, que ce soit dans une instruction SQL ou un test dans un programme. Ainsi, l'expression « nomComp='Air France' » n'aura pas la même signification que l'expression « nomComp='AIR France' ».

Premier exemple

Le tableau 1-2 décrit l'instruction SQL qui permet de créer, dans le schéma *soutou*, la table *vol_jour* illustrée par la figure suivante. L'absence du préfixe *soutou.* aurait conduit au même résultat si la connexion était établie par l'utilisateur *soutou* lors de la création de la table. L'utilisateur *soutou* devient propriétaire (*owner*) de l'objet table *vol_jour* (on dit aussi que le schéma *soutou* contient la table *vol_jour*).

Figure 1-1 Table à créer

VOL_JOUR					
NUM_VOL	AERO_DEP	AERO_ARR	COMP	JOUR_VOL	NB_PASSAGERS

Tableau 1-2 Création d'une table et de ses contraintes

Instruction SQL	Commentaires
<pre>CREATE TABLE vol_jour (num_vol VARCHAR2(6) NOT NULL, aero_dep VARCHAR2(3) NOT NULL, aero_arr VARCHAR2(3) NOT NULL, comp VARCHAR2(4) DEFAULT 'AF', jour_vol DATE NOT NULL, nb_passagers NUMBER(3));</pre>	<p>La table contient six colonnes (quatre chaînes de caractères variables, une date et un entier relatif de trois chiffres).</p> <p>La table inclut cinq contraintes en ligne.</p> <ul style="list-style-type: none"> • 4 NOT NULL qui imposent de renseigner quatre colonnes. • 1 DEFAULT qui fixe un code compagnie à défaut d'être renseigné.

Contraintes de colonnes

Les contraintes de colonnes ont pour but de programmer des règles de gestion au niveau des colonnes des tables. Elles peuvent alléger un développement côté client (si on déclare qu'une note doit être comprise entre 0 et 20, les programmes de saisie n'ont plus à tester les valeurs en entrée mais seulement le code retour après connexion à la base ; on déporte les contraintes côté serveur).

Les contraintes de colonnes peuvent être déclarées de deux manières :

- En même temps que la colonne (valable pour les contraintes monocolumnes), ces contraintes sont dites « en ligne » (*inline constraints*). L'exemple précédent en déclare deux.

- Une fois la colonne déclarée, ces contraintes ne sont pas limitées à une colonne et peuvent être personnalisées par un nom (*out-of-line constraints*).

En nommant chacune de vos contraintes de colonnes, vous disposez de quatre types possibles.

CONSTRAINT *nomContrainte*

- UNIQUE (*colonne1* [, *colonne2*]...)
 - PRIMARY KEY (*colonne1* [, *colonne2*]...)
 - FOREIGN KEY (*colonne1* [, *colonne2*]...)
 - REFERENCES [*schéma.*] *nomTablePere* (*colonne1* [, *colonne2*]...)
 - [ON DELETE { CASCADE | SET NULL }]
 - CHECK (*condition*)
- La contrainte UNIQUE impose une valeur distincte sur les colonnes concernées (les NULL font exception à moins que NOT NULL soit aussi appliqué sur chaque colonne).
 - La contrainte PRIMARY KEY déclare la clé primaire, qui impose une valeur distincte sur les colonnes concernées (NOT NULL est aussi appliqué sur chaque colonne).
 - La contrainte FOREIGN KEY déclare une clé étrangère pour relier cette table à une autre table père (voir la section « Intégrité référentielle » du chapitre 2).
 - La contrainte CHECK impose une condition simple ou complexe entre les colonnes de la table. Par exemple, CHECK(*nb_passagers*>0) interdira toute valeur négative tandis que CHECK(*aero_dep*!=*aero_arr*) interdira la saisie d'un trajet qui part et revient du même aéroport.



Dans le cas de UNIQUE et PRIMARY KEY, un index unique est généré sur les colonnes concernées. Vous pouvez disposer de plusieurs contraintes UNIQUE mais seule une clé primaire est autorisée.

Si vous ne nommez pas une de vos contraintes, un nom sera généré sous la forme suivante (figure 1-2 ci-contre).

Nous verrons au chapitre 3 comment ajouter, supprimer, désactiver, réactiver et différer des contraintes (options de la commande ALTER TABLE).

Conventions recommandées

Adoptez les conventions d'écriture suivantes pour vos contraintes :



- Préfixez par pk_ le nom d'une contrainte clé primaire, fk_ une clé étrangère, ck_ une vérification, un_ une unicité.
- Pour une contrainte clé primaire, suffixez du nom de la table la contrainte (exemple pk_Pilote).
- Pour une contrainte clé étrangère, renseignez (ou abréguez) les noms de la table source, de la clé, et de la table cible (exemple fk_Pil_compa_Comp).

Le script d'écriture des tables suivantes respecte ces conventions. La clé étrangère concrétise une association *un-à-plusieurs* entre les deux tables. Ici, il s'agit de relier chaque vol à sa compagnie (pour plus de détails concernant la modélisation, consultez la bibliographie « UML 2 pour les bases de données »).

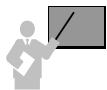
Tableau 1-3 Contraintes en ligne et nommées

Tables	Contraintes
<pre>CREATE TABLE compagnie (comp VARCHAR2(4), nom_comp VARCHAR2(15), date_creation DATE CONSTRAINT nn_date_crea NOT NULL, CONSTRAINT pk_compagnie PRIMARY KEY(comp), CONSTRAINT un_nom_comp UNIQUE(nom_comp));</pre>	<p>Une contrainte en ligne et deux contraintes hors ligne.</p>
<pre>CREATE TABLE vol_jour (num_vol VARCHAR2(6) NOT NULL, aero_dep VARCHAR2(3) CONSTRAINT nn_depart NOT NULL, aero_arr VARCHAR2(3) CONSTRAINT nn_arrivee NOT NULL, comp VARCHAR2(4) DEFAULT 'AF', jour_vol DATE NOT NULL, nb_passagers NUMBER(3), CONSTRAINT pk_vol_jour PRIMARY KEY(num_vol, jour_vol), CONSTRAINT fk_vol_jour_comp_compagnie FOREIGN KEY(comp) REFERENCES compagnie(comp), CONSTRAINT ck_nb_pax CHECK (nb_passagers>0), CONSTRAINT ck_trajet CHECK (aero_dep != aero_arr));</pre>	<p>Une contrainte en ligne nommée (NOT NULL) et quatre contraintes hors ligne nommées :</p> <ul style="list-style-type: none"> • Clé primaire. • CHECK (nombre d'heures de vol compris entre 0 et 20000). • UNIQUE (homonymes interdits). • Clé étrangère.

La figure suivante présente le détail des contraintes (capture d'écran de l'outil SQL Developer).

Figure 1-2 Contraintes d'une table

Colonne	Données	Contraintes	Droits	Statistiques	Déclencheurs	Flashback	Dépendances	Détails
1	2	3	4	5	6	7	8	9
CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION						
1 CK_NB_PAX	Check	nb_passagers>0						
2 CK_TRAJET	Check	aero_dep != aero_arr						
3 FK_VOL_JOUR_COMP_COMPAGNIE	Foreign_Key	(null)						
4 NN_ARRIVEE	Check	"AERO_ARR" IS NOT NULL						
5 NN_DEPART	Check	"AERO_DEP" IS NOT NULL						
6 PK_VOL_JOUR	Primary_Key	(null)						
7 SYS_C0010521	Check	"NUM_VOL" IS NOT NULL						
8 SYS_C0010524	Check	"JOUR_VOL" IS NOT NULL						



L'ordre de création des contraintes hors ligne n'est pas important au sein d'une table.

En revanche, l'ordre de création des tables est imposé, si les contraintes sont créées en même temps que les tables. En effet, il existe une certaine hiérarchie à respecter pour les clés étrangères : il faut d'abord créer les tables référentes, puis les tables qui en dépendent (la destruction des tables se fera dans l'ordre inverse).

Il est possible de créer les contraintes après avoir créé les tables via la commande `ALTER TABLE` (voir le chapitre 3).

Types des colonnes

Pour décrire les colonnes d'une table, Oracle fournit les types prédéfinis suivants (*built-in datatypes*) :

- caractères (CHAR, NCHAR, VARCHAR2, NVARCHAR2, CLOB, NCLOB, LONG) ;
- valeurs numériques NUMBER ;
- date/heure (DATE, INTERVAL DAY TO SECOND, INTERVAL YEAR TO MONTH, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE) ;
- données binaires (BLOB, BFILE, RAW, LONG RAW) ;
- adressage des enregistrements ROWID.

Détaillons à présent ces types. Nous verrons comment utiliser les plus courants au chapitre 2 et les autres au fil de l'ouvrage.

Caractères

Le tableau 1-4 décrit les types convenant aux données textuelles. NCHAR, NVARCHAR2 et NCLOB permettent de stocker des caractères Unicode (*multibyte*). Cette méthode de codage fournit une valeur unique pour chaque caractère quels que soient la plate-forme, le programme ou la langue.



Réservez le type CHAR aux données textuelles de taille fixe et constante.

Depuis Oracle 9, le type VARCHAR est remplacé par VARCHAR2. Le premier gère des chaînes maximales de 2 000 caractères et utilisait des NULL pour compléter chaque donnée. Le second est plus puissant en termes de stockage ; il n'occupe pas d'espace supplémentaire et n'utilise pas de NULL en interne.



Depuis la version 12c, la taille maximale d'un VARCHAR2 ou NVARCHAR2 peut être étendue à 32 767 octets (32 Ko) si le paramètre d'initialisation `MAX_STRING_SIZE` est positionné à `EXTENDED` (`STANDARD` par défaut). Une fois positionné, il ne vous sera plus possible de revenir à un comportement standard (limitation à 4 000 caractères).

Tableau 1-4 Types de données caractères

Type	Description	Commentaires pour une colonne
CHAR (<i>n</i> [BYTE CHAR])	Chaîne fixe de <i>n</i> caractères ou octets.	Taille fixe (complétée par des blancs si nécessaire). Maximum de 2 000 octets ou caractères.
VARCHAR2 (<i>n</i> [BYTE CHAR])	Chaîne variable de <i>n</i> caractères ou octets.	Taille variable. Maximum de 4 000 octets ou caractères.
NCHAR (<i>n</i>)	Chaîne fixe de <i>n</i> caractères Unicode.	Taille fixe (complétée par des blancs si nécessaire). Taille double pour le jeu AL16UTF16 et triple pour le jeu UTF8. Maximum de 2 000 caractères.
NVARCHAR2 (<i>n</i>)	Chaîne variable de <i>n</i> caractères Unicode.	Taille variable. Mêmes caractéristiques que NCHAR sauf pour la taille maximale qui est ici de 4 000 octets.
CLOB	Flot de caractères (CHAR).	Jusqu'à 4 gigaoctets.
NCLOB	Flot de caractères Unicode (NCHAR).	Idem CLOB.
LONG	Flot variable de caractères.	Jusqu'à 2 gigaoctets. Toujours fourni pour assurer la compatibilité, mais à remplacer par le type CLOB.
XMLTYPE	Stockage de documents XML	Jusqu'à 4 gigaoctets.

Valeurs numériques

Le type NUMBER sert à stocker des entiers positifs ou négatifs, des réels à virgule fixe ou flottante. La plage de valeurs possibles va de $\pm 1 \times 10^{-130}$ à $\pm 9.9...99 \times 10^{125}$ (trente-huit 9 suivis de quatre-vingt-huit 0).

Tableau 1-5 Type de données numériques

Type	Description	Commentaires pour une colonne
NUMBER (<i>t</i> , <i>d</i>)	Flottant de <i>t</i> chiffres dont <i>d</i> décimales.	Maximum pour <i>t</i> : 38. Plage pour <i>d</i> : [-84, +127]. Espace maximum utilisé : 21 octets.

Lorsque la valeur de *d* est négative, l'arrondi se réalise à gauche de la décimale comme le montre le tableau suivant.

Tableau 1-6 Représentation du nombre 7456123.89

Type	Description
NUMBER	7456123.89
NUMBER (9)	7456124
NUMBER (9, 2)	7456123.89
NUMBER (9, 1)	7456123.9
NUMBER (6)	Précision inférieure à la taille du nombre.
NUMBER (7, -2)	7456100
NUMBER (-7, 2)	Précision inférieure à la taille du nombre.



Déterminez toujours un nombre de décimales fixe de sorte à ne pas subir des arrondis, et donc des approximations, lors de calculs importants (sur des montants de facture ou des soldes de comptes bancaires, par exemple).

Pour définir des colonnes clé primaire, fixez toujours un nombre de décimales à zéro (par exemple, `NUMBER(3,0)` qui est identique à `NUMBER(3)`).

Enfin, n'utilisez pas toujours des entiers pour définir des clés primaire numériques, (par exemple, un numéro de Sécurité sociale `CHAR(13)` est préférable à `NUMBER(13)` car vous n'opérez jamais de calculs sur ces données, juste des tris ou des extractions de parties). De plus, si la taille de ce type de donnée n'est pas fixe, vous pourrez compléter avec des 0 devant (ce qui n'est pas possible pour les numériques).



Depuis la version 12c, il est possible d'utiliser un type numérique (entier) pour définir une colonne auto-incrémentée (voir le chapitre 2). Le mot-clé qui est utilisé dans les instructions `CREATE TABLE` et `ALTER TABLE` pour désigner un tel mécanisme est `GENERATED... AS IDENTITY...`

Flottants

Depuis Oracle 10g, deux types numériques apparaissent : `BINARY_FLOAT` et `BINARY_DOUBLE` qui permettent de représenter des grands nombres (plus importants que ceux définis par `NUMBER`) sous la forme de flottants. Les nombres flottants peuvent disposer d'une décimale située à tout endroit (de la première position à la dernière) ou ne pas avoir de décimale du tout. Un exposant peut éventuellement être utilisé (exemple : $1.777 e^{-20}$). Une échelle de valeurs ne peut être imposée à un flottant puisque le nombre de chiffres apparaissant après la décimale n'est pas restreint.

Tableau 1-7 Types de flottants

Type	Description	Commentaire pour une colonne
<code>BINARY_FLOAT</code>	Flottant simple précision.	Sur 5 octets (un représentant la longueur). Valeur entière maximale $3,4 \times 10^{+38}$, valeur entière minimale $-3,4 \times 10^{+38}$. Plus petite valeur positive $1,2 \times 10^{-38}$, plus petite valeur négative $-1,2 \times 10^{-38}$.
<code>BINARY_DOUBLE</code>	Flottant double précision.	Sur 9 octets (un représentant la longueur). Valeur entière maximale $1,79 \times 10^{+308}$, valeur entière minimale $-1,79 \times 10^{+308}$. Plus petite valeur positive $2,3 \times 10^{-308}$, plus petite valeur négative $-2,3 \times 10^{-308}$.

Le stockage des flottants diffère de celui des NUMBER en ce sens que le mécanisme de représentation interne est propre à Oracle. Pour une colonne NUMBER, les nombres à virgule ont une précision décimale. Pour les types BINARY_FLOAT et BINARY_DOUBLE, les nombres à virgule ont une précision exprimée en binaire.

Oracle fournit également le type ANSI FLOAT qui peut aussi s'écrire `FLOAT(n)`. L'entier *n* (de 1 à 126) indique la précision binaire. Afin de convertir une précision binaire en précision décimale, il convient de multiplier l'entier par 0.30103. La conversion inverse nécessite de multiplier *n* par 3.32193. Le maximum de 126 bits est à peu près équivalent à une précision de 38 décimales.

L'écriture d'un flottant est la suivante :

```
[+|-] {chiffre [chiffre]...[.] [chiffre [chiffre]...].chiffre [chiffre]...}
      [e[+|-] chiffre [chiffre]...] [f|d]
```

- e (ou E) indique la notation scientifique (mantisse et exposant) ;
- f (ou F) indique que le nombre est de type BINARY_FLOAT ;
- d (ou D) indique que le nombre est de type BINARY_DOUBLE.

Si le type n'est pas explicitement précisé, l'expression est considérée comme de type NUMBER.

Date/heure

- Le type DATE permet de stocker des moments ponctuels, la précision est composée du siècle, de l'année, du mois, du jour, de l'heure, des minutes et des secondes.
- Le type TIMESTAMP est plus précis dans la définition d'un moment (fraction de seconde).
- Le type TIMESTAMP WITH TIME ZONE prend en compte les fuseaux horaires.
- Le type TIMESTAMP WITH LOCAL TIME ZONE permet de faire la dichotomie entre une heure côté serveur et une heure côté client.
- Le type INTERVAL YEAR TO MONTH permet d'extraire une différence entre deux moments avec une précision mois/année.
- Le type INTERVAL DAY TO SECOND permet d'extraire une différence plus précise entre deux moments (précision de l'ordre de la fraction de seconde).

Tableau 1-8 Types de données date/heure

Type	Description	Commentaires pour une colonne
DATE	Date et heure du 1 ^{er} janvier 4712 avant J.-C. au 31 décembre 4712 après J.-C.	Sur 7 octets. Le format par défaut est spécifié par le paramètre <code>NLS_DATE_FORMAT</code> .
INTERVAL YEAR (an) TO MONTH	Période représentée en années et mois.	Sur 5 octets. La précision de <i>an</i> va de 0 à 9 (par défaut 2).
INTERVAL DAY (jo) TO SECOND (fsec)	Période représentée en jours, heures, minutes et secondes.	Sur 11 octets. Les précisions <i>jo</i> et <i>fsec</i> vont de 0 à 9 (par défaut 2 pour le jour et 6 pour les fractions de secondes).
TIMESTAMP (fsec)	Date et heure incluant des fractions de secondes (précision qui dépend du système d'exploitation).	De 7 à 11 octets. La valeur par défaut du paramètre d'initialisation est située dans <code>NLS_TIMESTAMP_FORMAT</code> . La précision des fractions de secondes va de 0 à 9 (par défaut 6).
TIMESTAMP (fsec) WITH TIME ZONE	Date et heure avec le décalage de Greenwich (UTC) au format ' <i>h:m</i> ' (<i>heures:minutes</i> par rapport au méridien, exemple : '-5:0').	Sur 13 octets. La valeur par défaut du paramètre de l'heure du serveur est située dans <code>NLS_TIMESTAMP_TZ_FORMAT</code> .
TIMESTAMP (fsec) WITH LOCAL TIME ZONE	Comme le précédent mais cadré sur l'heure locale (client) qui peut être différente de celle du serveur.	De 7 à 11 octets.



N'utilisez jamais un format textuel pour stocker des dates ou des heures (par exemple, `CHAR(10)` pour un format *jj/mm/aaaa*) car vous ne pourrez pas bénéficier de contrôles et de calculs, toujours nécessaires à un moment donné dans ces cas-là.

Données binaires

Le tableau 1-9 présente les types permettant de stocker des données non structurées (images, sons, etc.).

Tableau 1-9 Types de données binaires

Type	Description	Commentaires pour une colonne
BLOB	Données binaires non structurées.	Jusqu'à 4 gigaoctets.
BFILE	Données binaires stockées dans un fichier externe à la base.	idem.
RAW(<i>size</i>)	Données binaires.	Jusqu'à 2 000 octets.
LONG RAW	Données binaires.	Jusqu'à 2 gigaoctets, toujours fourni pour assurer la compatibilité, mais à remplacer par le type <code>BLOB</code> .

Structure d'une table (DESC)

DESC (raccourci de DESCRIBE) est une commande SQL*Plus, car elle n'est comprise que dans l'interface de commandes d'Oracle. Elle permet d'extraire la structure brute d'une table. Elle peut aussi s'appliquer à une vue ou un synonyme. Enfin, elle révèle également les paramètres d'une fonction ou procédure cataloguée.

■ **DESC** [RIBE] [*schéma.*] *élément*

Si le schéma n'est pas indiqué, il s'agit de celui de l'utilisateur connecté. L'élément désigne le nom d'une table, vue, procédure, fonction ou synonyme.

La structure brute des tables précédemment créées est présentée à l'aide des commandes suivantes. Il n'y a que le nom, le type et la non-nullité de la colonne qui apparaissent. Le nom des contraintes n'est pas indiqué ici (comme peut le produire l'outil SQL Developer, voir figure 1-2).

Figure 1-3 Structure brute des tables

```
SQL> desc vol_jour
Nom                                NULL ?   Type
-----
NUM_UOL                            NOT NULL VARCHAR2(6)
AERO_DEP                            NOT NULL VARCHAR2(3)
AERO_ARR                            NOT NULL VARCHAR2(3)
COMP                                VARCHAR2(4)
JOUR_UOL                            NOT NULL DATE
NB_PASSAGERS                        NUMBER(3)

SQL> desc compagnie
Nom                                NULL ?   Type
-----
COMP                                NOT NULL VARCHAR2(4)
NOM_COMP                            VARCHAR2(15)
DATE_CREATION                       NOT NULL DATE
```

Commentaires stockés (COMMENT)

Les commentaires stockés permettent de documenter une table, une colonne ou une vue. L'instruction SQL pour créer un commentaire est COMMENT.

■ **COMMENT** ON { TABLE [*schéma.*]*nomTable* |
COLUMN [*schéma.*]*nomTable.nomColonne* }
IS '*Texte décrivant le commentaire*';

Pour supprimer un commentaire, il suffit de le redéfinir en inscrivant une chaîne vide (' ') dans la clause `IS`. Une fois définis, nous verrons à la section « Dictionnaire des données » du chapitre 5 comment retrouver ces commentaires.

Le premier commentaire du script ci-après documente la table `Compagnie`, les trois suivants renseignent trois colonnes de cette table. La dernière instruction supprime le commentaire à propos de la colonne `nomComp`.

```
COMMENT ON TABLE Compagnie IS 'Table des compagnies aériennes françaises';
COMMENT ON COLUMN Compagnie.comp IS 'Code abrégé de la compagnie';
COMMENT ON COLUMN Compagnie.nomComp IS 'Un mauvais commentaire';
COMMENT ON COLUMN Compagnie.ville IS 'Ville de la compagnie, défaut :
Paris';
COMMENT ON COLUMN Compagnie.nomComp IS '';
```

Noms des objets

Chaque objet ou constituant de la base (table, index, contrainte, colonne, variable, etc.) est nommé à l'aide d'un identifiant de 1 à 30 caractères (composé de lettres, de chiffres ou des caractères `_`, `$` et `#`).

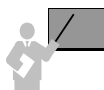
Le nom peut être écrit entre guillemets – la casse doit alors être impérativement respectée de même que l'utilisation des guillemets (on parle de *quoted identifier*). Le seul exemple présenté dans cet ouvrage qui adopte ce style de notation est le suivant ; vous y découvrirez la possibilité de décrire des identifiants sous la forme de mots séparés par des espaces (ce qui n'est pas conseillé).

Figure 1-4 Nommage de type « quoted identifier »

```
SQL> CREATE TABLE "vols du jour"
  2 ("numVol" VARCHAR2(6) NOT NULL,
  3 "comp" VARCHAR2(4) DEFAULT 'AF',
  4 "jour vol" DATE NOT NULL,
  5 CONSTRAINT "pk vols du jour" PRIMARY KEY("numVol", "jour vol"));

Table cr  e.
SQL> DESC "vols du jour"
Nom                                     NULL ?   Type
-----
numVol                                  NOT NULL VARCHAR2(6)
comp                                     VARCHAR2(4)
jour vol                                 NOT NULL DATE

SQL> SELECT numVol FROM "vols du jour";
      *
ERREUR   la ligne 1 :
ORA-00904: "NUMVOL" : identificateur non valide
```



Seuls les noms de base (8 caractères) et les noms de *database link* (128 caractères) sont toujours stockés en majuscules et sont insensibles à la casse.

Le nom de chaque colonne doit être unique pour une table (en revanche, le même nom d'une colonne peut être utilisé dans différentes tables). Les noms des objets (tables, colonnes, contraintes, vues, etc.) doivent être uniques au niveau du schéma (en revanche, plusieurs tables peuvent porter le même nom dans différents schémas).

Avec une notation sans guillemets, vous ne devez pas emprunter les mots réservés (TABLE, SELECT, INSERT, IF, etc.). Vous trouverez la liste de ces mots réservés dans la documentation officielle à l'annexe D du livre *SQL Reference*.

Il n'est pas recommandé d'utiliser les caractères \$ et # (très employés en interne par Oracle).

Les identifiants sans guillemets (*nonquoted identifiers*) ne sont donc pas sensibles à la casse et sont traduits en majuscules dans le dictionnaire des données (voir le chapitre 5). Ainsi, les trois écritures désignent le même identifiant : `aeroport`, `AEROPORT` et `"AEROPORT"`.

Utilisation de SQL Developer Data Modeler

Une des utilisations de l'outil d'Oracle SQL Developer Data Modeler consiste à générer des scripts de création de tables (DDL scripts, DDL pour *Data Definition Language*) après avoir saisi les caractéristiques de chaque table sous une forme graphique (modèle relationnel des données). Ce procédé est appelé *forward engineering* car il chemine dans le sens d'une conception classique pour laquelle l'étape finale est concrétisée par la création des tables.

Dans l'arborescence de gauche, par un clic droit sur l'élément `Modèles relationnels`, choisir `Nouveau modèle relationnel`. Une fois dans le modèle relationnel, les icônes indiquées vous permettront de créer vos tables et toutes les liaisons entre elles (clés étrangères). À titre d'exemple, créons deux tables reliées par une clé étrangère (ici, un pilote qui est rattaché à sa compagnie).

Pour créer une table, vous devez la nommer dans la fenêtre de saisie (le choix `Appliquer` modifiera le nom complet), puis définir ses colonnes. En choisissant l'entrée `Colonnes`, le symbole « + » vous permettra de saisir le nom et le type de chaque colonne de la table. N'ajoutez aucune contrainte pour l'instant (clé primaire et clé étrangère), contentez-vous de saisir les colonnes sans ajouter de colonnes de nature clé étrangère.

Figure 1-5 Création d'un modèle relationnel avec Data Modeler

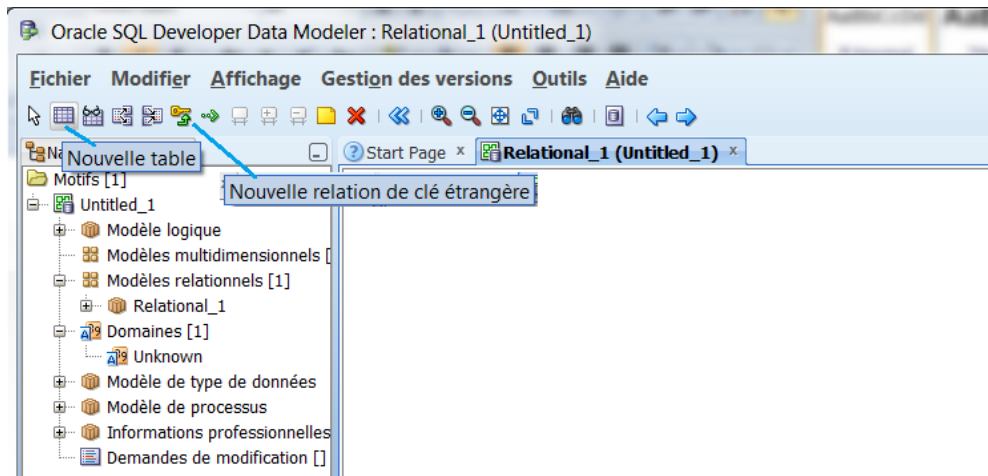
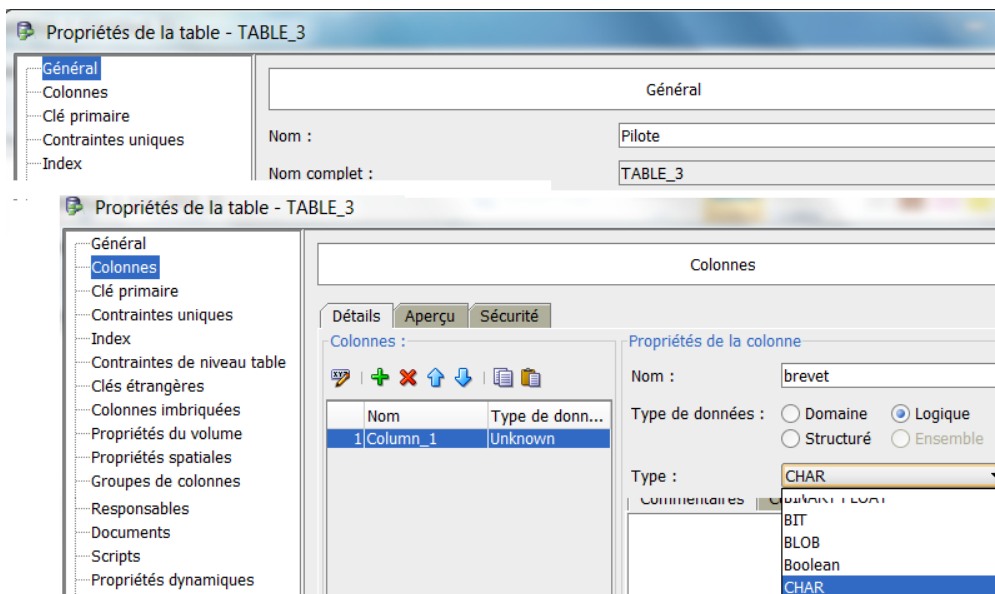
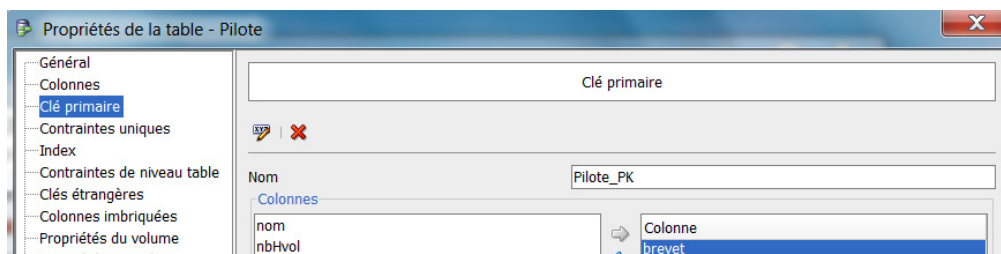


Figure 1-6 Colonnes d'une table avec Data Modeler



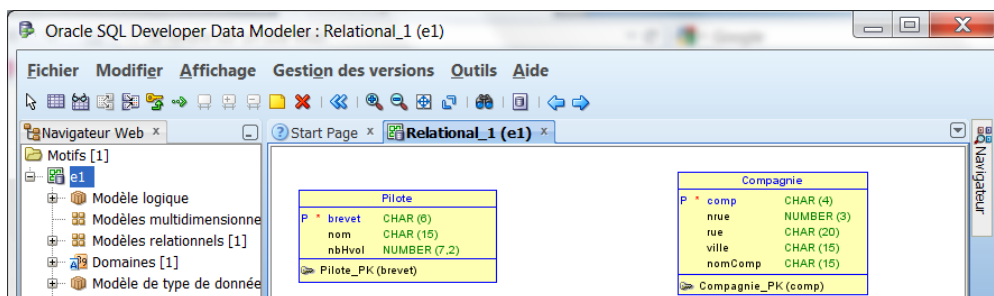
Définissez ensuite la clé primaire de chaque table (colonne brevet de la table Pilote et colonne comp de la table Compagnie).

Figure 1-7 Clé primaire avec Data Modeler



À l'issue de cette étape, le diagramme se modifie pour faire apparaître les deux nouvelles contraintes.

Figure 1-8 Tables dotées d'une clé primaire avec Data Modeler



Reliez la table Compagnie à la table Pilote en sélectionnant l'icône de clé étrangère. Une boîte de dialogue s'affiche alors et décrit les caractéristiques de la nouvelle contrainte référentielle (voir le chapitre 3).



Assurez-vous que la table source corresponde la table de r f f e n c e (ici, les compagnies) et n interpr tez pas le terme `source` comme source de la fl che qui d crit le sens du lien, et `cible` comme la table cible de ce lien. C est tout l inverse

Vous remarquerez que la colonne de type clé étrangère générée est automatiquement nommée à l'aide de la table et de la colonne de référence : `tablesource_cleprimaire` (ici, `Compagnie_comp`, que nous choisissons de renommer `compa`).

Figure 1-9 Définition d'une clé étrangère avec Data Modeler

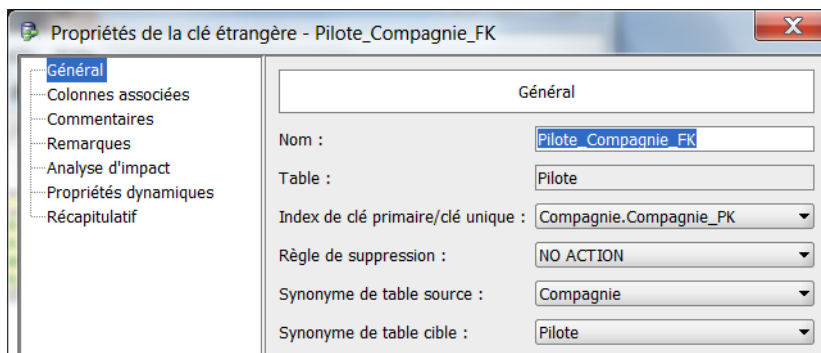
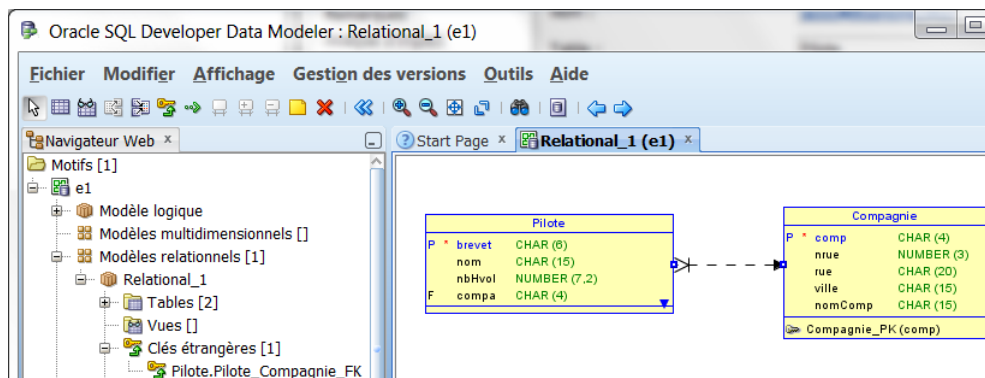
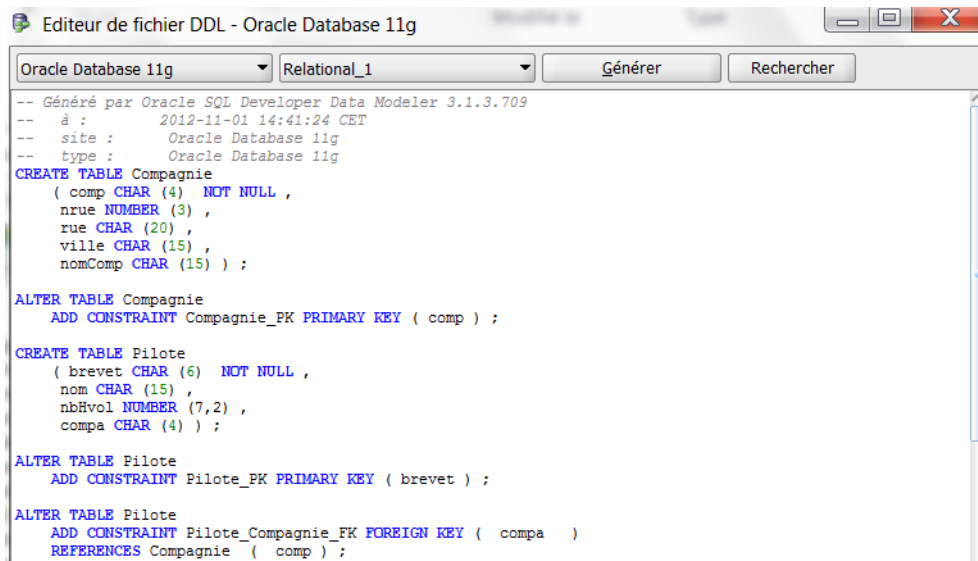


Figure 1-10 Modèle relationnel final avec Data Modeler



La génération du script SQL s'opère par le menu Fichier/Exporter/Fichier DDL. Sélectionner la version du SGBD cible, puis choisir Générer. Tous les éléments du modèle relationnel sont sélectionnés par défaut, mais vous pouvez volontairement écarter certaines tables du script. Une fois votre sélection faite, le script SQL se génère automatiquement. Vous noterez que les contraintes sont déclarées après les tables (voir la commande ALTER TABLE au chapitre 3). Ce procédé est bien adapté à la majorité des outils de conception, qui l'adoptent pour leur processus de *reverse engineering*.

Figure 1-11 Script de génération des tables avec Data Modeler



```
Editeur de fichier DDL - Oracle Database 11g
Oracle Database 11g Relational_1 Générer Rechercher
-- Généré par Oracle SQL Developer Data Modeler 3.1.3.709
-- à : 2012-11-01 14:41:24 CET
-- site : Oracle Database 11g
-- type : Oracle Database 11g
CREATE TABLE Compagnie
( comp CHAR (4) NOT NULL ,
  nrue NUMBER (3) ,
  rue CHAR (20) ,
  ville CHAR (15) ,
  nomComp CHAR (15) ) ;

ALTER TABLE Compagnie
  ADD CONSTRAINT Compagnie_PK PRIMARY KEY ( comp ) ;

CREATE TABLE Pilote
( brevet CHAR (6) NOT NULL ,
  nom CHAR (15) ,
  nbHvol NUMBER (7,2) ,
  compa CHAR (4) ) ;

ALTER TABLE Pilote
  ADD CONSTRAINT Pilote_PK PRIMARY KEY ( brevet ) ;

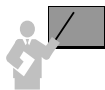
ALTER TABLE Pilote
  ADD CONSTRAINT Pilote_Compagnie_FK FOREIGN KEY ( compa )
  REFERENCES Compagnie ( comp ) ;
```

Suppression des tables

Il vous sera sans doute utile d'écrire un script qui supprime tout ou partie des tables de votre schéma. Ainsi, vous pourrez recréer un ensemble homogène de tables (comme une sorte de base « vierge ») à la demande. Bien entendu, si des données sont présentes dans vos tables, vous devrez opter pour une stratégie d'exportation ou de sauvegarde avant de réinjecter vos données dans les nouvelles tables. À ce stade de la lecture de l'ouvrage, vous n'en êtes pas là, et le script de suppression vous permettra de corriger les erreurs de syntaxe que vous risquez fort de faire lors de l'écriture du script de création des tables.

Si vous définissez des contraintes en même temps que les tables (dans l'ordre CREATE TABLE...), vous devrez respecter l'ordre suivant : tables « pères » (de référence), puis les tables « fils » (dépendantes). L'ordre de suppression des tables, pour des raisons de cohérence, est totalement inverse : vous devez supprimer les tables « fils » d'abord, puis les tables de référence. Dans l'exemple présenté à la section « Conventions recommandées », il serait malvenu de vouloir supprimer la table *Compagnie* avant de supprimer la table *Pilote*. En effet, la clé étrangère *compa* n'aurait plus de sens. Cela n'est d'ailleurs pas possible sans forcer l'option `CASCADE CONSTRAINTS` (voir plus loin).

```
DROP TABLE [schéma.]nomTable [CASCADE CONSTRAINTS] [PURGE];
```



- Pour pouvoir supprimer une table dans son schéma, il faut que la table appartienne à l'utilisateur. Si l'utilisateur a le privilège `DROP ANY TABLE`, il peut supprimer une table dans tout schéma.
- L'instruction `DROP TABLE` entraîne la suppression des données, de la structure, de la description dans le dictionnaire des données, des index, des déclencheurs associés (*triggers*) et la récupération de la place dans l'espace de stockage.

- `CASCADE CONSTRAINTS` permet de s'affranchir des clés étrangères actives contenues dans d'autres tables et qui référencent la table à supprimer. Cette option détruit les contraintes des tables « fils » associées sans rien modifier aux données qui y sont stockées (voir section « Intégrité référentielle » du prochain chapitre).
- `PURGE` permet de récupérer instantanément l'espace alloué aux données de la table (les blocs de données) sans les disposer dans la poubelle d'Oracle (*recycle bin*) .

Certains éléments qui utilisaient la table (vues, synonymes, fonctions ou procédures) ne sont pas supprimés mais sont temporairement inopérants. En revanche, les éventuels index et déclencheurs sont supprimés.



- Une suppression (avec `PURGE`) ne peut pas être annulée par la suite.
- La suppression d'une table sans `PURGE` peut être récupérée via l'espace *recycle bin* par la technologie *flashback* (ce mécanisme, qui relève davantage de l'administration, sort du cadre de cet ouvrage).



Si les contraintes sont déclarées au sein des tables (dans chaque instruction `CREATE TABLE`), il vous suffit de relire à l'envers le script de création des tables pour en déduire l'ordre de suppression.

Utilisez avec parcimonie l'option `CASCADE CONSTRAINTS` qui fera fi, sans vous le dire, du mécanisme de l'intégrité référentielle assuré par les clés étrangères (voir le chapitre 3).

Exercices

L'objectif de ces exercices est de créer des tables, leur clé primaire et des contraintes de vérification (NOT NULL et CHECK). La première partie des exercices (de 1.1 à 1.4) concerne la base *Parc Informatique*. Le dernier exercice traite d'une autre base (*Chantiers*) que vous pouvez appliquer à la version d'Oracle à partir de la 11g.

Exercice 1.1 Présentation de la base de données

Une entreprise désire gérer son parc informatique à l'aide d'une base de données. Le bâtiment est composé de trois étages. Chaque étage possède son réseau (ou segment distinct) Ethernet. Ces réseaux traversent des salles équipées de postes de travail. Un poste de travail est une machine sur laquelle sont installés certains logiciels. Quatre catégories de postes de travail sont recensées (stations Unix, terminaux X, PC Windows et PC NT). La base de données devra aussi décrire les installations de logiciels.

Les noms et types des colonnes sont les suivants :

Tableau 1-10 Caractéristiques des colonnes

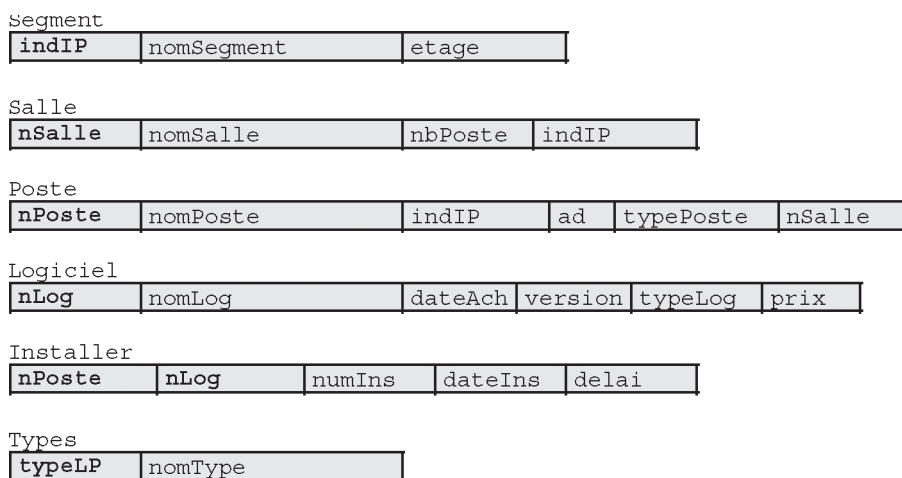
Colonne	Commentaires	Types
indIP	Trois premiers groupes IP (exemple : 130.120.80).	VARCHAR2 (11)
nomSegment	Nom du segment.	VARCHAR2 (20)
etage	Étage du segment.	NUMBER (2)
nSalle	Numéro de la salle.	VARCHAR2 (7)
nomSalle	Nom de la salle.	VARCHAR2 (20)
nbPoste	Nombre de postes de travail dans la salle.	NUMBER (2)
nPoste	Code du poste de travail.	VARCHAR2 (7)
nomPoste	Nom du poste de travail.	VARCHAR2 (20)
ad	Dernier groupe de chiffres IP (exemple : 11).	VARCHAR2 (3)
typePoste	Type du poste (Unix, TX, PCWS, PCNT).	VARCHAR2 (9)
dateIns	Date d'installation du logiciel sur le poste.	DATE
nLog	Code du logiciel.	VARCHAR2 (5)
nomLog	Nom du logiciel.	VARCHAR2 (20)
dateAch	Date d'achat du logiciel.	DATE
version	Version du logiciel.	VARCHAR2 (7)
typeLog	Type du logiciel (Unix, TX, PCWS, PCNT).	VARCHAR2 (9)
prix	Prix du logiciel.	NUMBER (6, 2)
numIns	Numéro séquentiel des installations.	NUMBER (5)
dateIns	Date d'installation du logiciel.	DATE
delai	Intervalle entre achat et installation.	INTERVAL DAY (5) TO SECOND (2) ,
typeLP	Types des logiciels et des postes.	VARCHAR2 (9)
nomType	Noms des types (Terminaux X, PC Windows...).	VARCHAR2 (20)

Exercice 1.2 Création des tables

Écrivez puis exécutez le script SQL (que vous appellerez `creParc.sql`) de création des tables avec leur clé primaire (en gras dans le schéma suivant) et les contraintes suivantes :

- Les noms des segments, des salles et des postes sont non nuls.
- Le domaine de valeurs de la colonne `ad` s'étend de 0 à 255.
- La colonne `prix` est supérieure ou égale à 0.
- La colonne `dateIns` est égale à la date du jour par défaut.

Figure 1-12 Schéma des tables



Exercice 1.3 Structure des tables

Écrivez puis exécutez le script SQL (que vous appellerez `descParc.sql`) qui affiche la description de toutes ces tables (en utilisant des commandes `DESC`). Comparez avec le schéma.

Exercice 1.4 Destruction des tables

Écrivez puis exécutez le script SQL de destruction des tables (que vous appellerez `dropParc.sql`). Lancez ce script puis à nouveau celui de la création des tables.

Exercice 1.5 Schéma de la base *Chantiers* (Oracle 11g)

Une société désire informatiser les visites des chantiers de ses employés. Pour définir cette base de données, une première étude fait apparaître les informations suivantes :

- Chaque employé est modélisé par un numéro, un nom et une qualification.
- Un chantier est caractérisé par un numéro, un nom et une adresse.
- L'entreprise dispose de véhicules pour lesquels il est important de stocker pour le numéro d'immatriculation, le type (un code valant par exemple 0 pour une camionnette, 1 pour une moto et 2 pour une voiture) ainsi que le kilométrage en fin d'année.
- Le gestionnaire a besoin de connaître les distances parcourues par un véhicule pour chaque visite d'un chantier.
- Chaque jour, un seul employé sera désigné conducteur des visites d'un véhicule.
- Pour chaque visite, il est important de pouvoir connaître les employés transportés.

Les colonnes à utiliser sont les suivantes :

Tableau 1-11 Caractéristiques des colonnes à ajouter

Colonne	Commentaires	Types
kilometres	Kilométrage d'un véhicule lors d'une sortie.	NUMBER
n_conducteur	Numéro de l'employé conducteur.	VARCHAR2 (4)
n_transporte	Numéro de l'employé transporté.	VARCHAR2 (4)

L'exercice consiste à compléter le schéma relationnel ci-après (ajout de colonnes et définition des contraintes de clé primaire et étrangère).

```
CREATE TABLE Employe (n_emp VARCHAR(4), nom_emp VARCHAR(20),
    qualif_emp VARCHAR(12), CONSTRAINT pk_emp PRIMARY KEY(n_emp));

CREATE TABLE Chantier (n_chantier VARCHAR(10), nom_ch VARCHAR(10),
    adresse_ch VARCHAR(15), CONSTRAINT pk_chan PRIMARY KEY(n_chantier));

CREATE TABLE Vehicule (n_vehicule VARCHAR(10), type_vehicule VARCHAR(1),
    kilometrage NUMBER, CONSTRAINT pk_vehi PRIMARY KEY(n_vehicule));

CREATE TABLE Visite(n_chantier VARCHAR(10), n_vehicule VARCHAR(10),
    date_jour DATE, ...
    CONSTRAINT pk_visite PRIMARY KEY(...),
    CONSTRAINT fk_depl_chantier FOREIGN KEY(n_chantier) ...,
    CONSTRAINT fk_depl_vehicule FOREIGN KEY(n_vehicule) ...,
    CONSTRAINT fk_depl_employe FOREIGN KEY(n_conducteur) ... );

CREATE TABLE Transporter (...
    CONSTRAINT pk_transporter PRIMARY KEY (...),
    CONSTRAINT fk_transp_visite FOREIGN KEY ... ,
    CONSTRAINT fk_transp_employe FOREIGN KEY ...);
```


Chapitre 2

Manipulation des données

Ce chapitre décrit une partie de l'aspect DML (*Data Manipulation Language*) du langage SQL d'Oracle. Bien qu'il existe d'autres possibilités d'insérer des données (techniques d'importation ou de chargement), SQL propose trois instructions de base pour manipuler des données :

- l'insertion d'enregistrements : `INSERT` ;
- la modification de données : `UPDATE` ;
- la suppression d'enregistrements : `DELETE`.

Insertions d'enregistrements (INSERT)

Pour pouvoir insérer des enregistrements dans une table, il faut que cette dernière soit dans votre schéma ou que vous ayez reçu le privilège `INSERT` sur la table. Si vous avez le privilège `INSERT ANY TABLE`, vous pouvez ajouter des données dans n'importe quelle table de tout schéma.

Il existe plusieurs possibilités d'insertion : l'insertion monoligne qui ajoute un enregistrement par instruction (que nous allons détailler maintenant) et l'insertion multiligne qui insère plusieurs valeurs (que nous détaillerons au chapitre 4).

Syntaxe

La syntaxe simplifiée de l'instruction `INSERT` monoligne est la suivante :

```
INSERT INTO [schéma.] { nomTable | nomVue | requêteSELECT }  
    [(colonne1, colonne2...)]  
VALUES (valeur1 | DEFAULT, valeur2 | DEFAULT...);
```

À l'aide d'exemples, nous allons détailler les possibilités de cette instruction en considérant la majeure partie des types de données proposés par Oracle.

Renseigner ou pas toutes les colonnes

Le script suivant insère trois compagnies et quatre vols en utilisant différentes options de l'instruction `INSERT`. Il s'agit de renseigner ou pas les colonnes d'une table par une liste de valeurs de type adéquat. Le mot-clé `DEFAULT` utilisé en tant que valeur permet d'affecter explicitement une valeur par défaut à la colonne associée.

Tableau 2-1 Insertions de lignes

Instructions SQL	Commentaires
<pre>INSERT INTO compagnie VALUES ('SING', 'Singapore AL', TO_ DATE('19470101', 'YYYYMMDD'));</pre>	Les valeurs sont renseignées dans l'ordre de la structure de la table.
<pre>INSERT INTO compagnie (nom_comp, comp, date_creation) VALUES ('Air France', 'AF', TO_DATE('19330101', 'YYYYMMDD'));</pre>	Les valeurs sont renseignées dans l'ordre de la liste.
<pre>INSERT INTO compagnie (nom_comp, comp, date_creation) VALUES (NULL, 'GO', TO_DATE('20141231', 'YYYYMMDD'));</pre>	
<pre>INSERT INTO vol_jour (num_vol, aero_dep, aero_arr, jour_vol, nb_passagers) VALUES ('AF6143', 'TLS', 'ORY', TO_DATE('20141120 15:30', 'YYYYMMDD HH24:MI'), 120);</pre>	La compagnie est omise, donc la valeur par défaut s'appliquera (AF).
<pre>INSERT INTO vol_jour (num_vol, aero_dep, aero_arr, jour_vol) VALUES ('AF6145', 'ORY', 'TLS', TO_DATE('20141120 18:45', 'YYYYMMDD HH24:MI'));</pre>	Le nombre de passagers est omis, donc <code>NULL</code> s'appliquera.
<pre>INSERT INTO vol_jour (num_vol, aero_dep, aero_arr, comp, jour_vol, nb_passagers) VALUES ('SQ747', 'CDG', 'SIN', 'SING', TO_DATE('20141120 19:30', 'YYYYMMDD HH24:MI'), NULL);</pre>	Le nombre de passagers est explicitement valué à <code>NULL</code> .
<pre>INSERT INTO vol_jour (num_vol, aero_dep, aero_arr, comp, jour_vol, nb_ passagers) VALUES ('AF6550', 'CDG', 'TLS', DEFAULT, TO_DATE('20141120 20:00', 'YYYYMMDDHH24:MI'), 195);</pre>	La compagnie est explicitement valuée à la valeur par défaut.



Bannissez le premier style d'écriture et renseignez toujours le plus de colonnes possible dans vos instructions `INSERT`, vous subirez ainsi le moins de comportements par défaut.

Une fois la validation effectuée (par `commit`, voir chapitre 6), le résultat est présenté avec SQL Developer de la manière suivante.

Figure 2-1 Tables après les insertions

COMPAGNIE		
COMP	NOM_COMP	DATE_CREATION
SING	Singapore AL	01/01/47
AF	Air France	01/01/33
GO	(null)	31/12/14

VOL_JOUR					
NUM_VOL	AERO_DEP	AERO_ARR	COMP	JOUR_VOL	NB_PASSAGERS
AF6143	TLS	ORY	AF	20/11/14	120
AF6550	CDG	TLS	AF	20/11/14	195
SQ747	CDG	SIN	SING	20/11/14	(null)
AF6145	ORY	TLS	AF	20/11/14	(null)



Depuis la version 12c, toute colonne peut être définie à `DEFAULT ON NULL valeur_defaut`. Ainsi, l'insertion d'un `NULL`, qu'elle soit explicite ou implicite, sera remplacée par la valeur par défaut.

Ne pas respecter des contraintes

Insérons des vols qui ne respectent pas des contraintes. Les messages renvoyés pour chaque erreur font apparaître le nom de la contrainte. Les valeurs qui sont les facteurs déclenchant sont notées en gras. La première erreur est un doublon de clé primaire, la deuxième un `NULL` interdit et la troisième une condition de vérification. La dernière erreur signifie que la clé étrangère référence un absent (pour plus de détails à ce sujet, consultez la section « Intégrité référentielle »).

Tableau 2-2 Erreur typique de contraintes

Instruction	Message d'erreur
<pre>SQL> INSERT INTO vol_jour (num_vol,aero_dep,aero_arr,comp,jour_vol, nb_passagers) VALUES ('AF6550', 'AGN', 'TLS', 'AF', TO_DATE('20141120 20:00', 'YYYYMMDD HH24:MI'),95);</pre>	ERREUR à la ligne 1 : ORA-00001: violation de contrainte unique (SOUTOU.PK_VOL_ JOUR)
<pre>SQL> INSERT INTO vol_jour (num_vol,aero_dep,aero_arr,comp,jour_vol, nb_passagers) VALUES ('AF6530', 'AGN', NULL, 'AF', TO_DATE('20141120 10:00', 'YYYYMMDD HH24:MI'),95);</pre>	ERREUR à la ligne 3 : ORA-01400: impossible d'insérer NULL dans ("SOU- TOU"."VOL_ JOUR"."AERO_ARR")
<pre>SQL> INSERT INTO vol_jour (num_vol,aero_dep,aero_arr,comp,jour_vol, nb_passagers) VALUES ('AF6530', 'AGN', 'AGN', 'AF', TO_DATE('20141120 10:00', 'YYYYMMDD HH24:MI'),95);</pre>	ERREUR à la ligne 1 : ORA-02290: violation de contraintes (SOUTOU.CK_TRAJET) de vérification
<pre>SQL> INSERT INTO vol_jour (num_vol,aero_dep,aero_arr,comp,jour_vol, nb_passagers) VALUES ('AF6530', 'AGN', 'TLS', 'BA', TO_DATE('20141120 10:00', 'YYYYMMDD HH24:MI'),95);</pre>	ERREUR à la ligne 1 : ORA-02291: violation de contrainte d'intégrité (SOUTOU.FK_VOL_ JOUR_COMP_COMPA- GNIE) - clé parent introuvable

Dates/heures

Nous avons décrit au chapitre 1 les caractéristiques générales des types Oracle pour stocker des éléments de type date/heure.

Type DATE

Déclarons la table `Pilote` qui contient deux colonnes de type `DATE`.

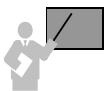
```
CREATE TABLE pilote
(brevet   VARCHAR2(6),          prenom   VARCHAR2(20) NOT NULL,
 nom      VARCHAR2(20) NOT NULL, date_nais DATE NOT NULL,
 embauche DATE NOT NULL,
 CONSTRAINT pk_pilote PRIMARY KEY(brevet));
```

La première insertion initialise la date de naissance au 5 février 1965 (à zéro heure, zéro minute et zéro seconde), tandis que la date d'embauche inclura les heures, minutes et secondes

par la fonction `SYSDATE`. La seconde insertion utilise un autre format en entrée et initialisera l'embauche au jour présent (à zéro heure, zéro minute et zéro seconde) par la fonction `TRUNC`.

```
INSERT INTO Pilote(brevet, prenom, nom, date_nais, embauche)
VALUES ('B1', 'Christian', 'Mermoz',
        TO_DATE('05/02/1965', 'DD/MM/YYYY'), SYSDATE);

INSERT INTO Pilote(brevet, prenom, nom, date_nais, embauche)
VALUES ('B2', 'Christian', 'Mermoz',
        TO_DATE('19650205', 'YYYYMMDD'), TRUNC(SYSDATE));
```



La fonction `TO_DATE` doit toujours être utilisée pour appliquer un format à la date qui peut être précis à la seconde. Par exemple, le 5 février 1965 à 6 h 30 sera codé `TO_DATE('05-02-1965:06:30', 'DD-MM-YYYY:HH24:MI')`.

Nous verrons au chapitre 4 comment afficher les heures, minutes et secondes d'une colonne de type `DATE`. Nous verrons aussi qu'il est possible d'ajouter ou de soustraire des dates entre elles.

Types `TIMESTAMP`

La table `Evenements` contient la colonne `arrive` (`TIMESTAMP`) pour stocker des fractions de secondes et la colonne `arriveLocalement` (`TIMESTAMP WITH TIME ZONE`) pour considérer aussi le fuseau horaire.

```
CREATE TABLE Evenements
(arrive TIMESTAMP, arriveLocalement TIMESTAMP WITH TIME ZONE);
```

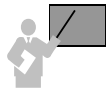
L'insertion suivante initialise :

- la colonne `arrive` au 5 février 1965 à 9 heures, 30 minutes, 2 secondes et 123 centièmes dans le fuseau défini au niveau de la base ;
- la colonne `arriveLocalement` au 16 janvier 1965 à 12 heures, 30 minutes, 5 secondes et 98 centièmes dans le fuseau décalé vers l'est de 4 h 30 par rapport au méridien de Greenwich.

```
INSERT INTO Evenements(arrive, arriveLocalement)
VALUES (TIMESTAMP '1965-02-05 09:30:02.123',
        TIMESTAMP '1965-01-16 12:30:05.98 + 4:30');
```

Le format par défaut de ces types est décrit dans les variables `NLS_TIMESTAMP_FORMAT` ('YYYY-MM-DD HH:MM:SS.d' d : décimales) et `NLS_TIMESTAMP_TZ_FORMAT` ('YYYY-MM-DD HH:MM:SS.d ± hh:mn', avec hh:mn en heures-minutes par rapport à Greenwich).

Types *INTERVAL*



Les types *INTERVAL* permettent de déclarer des durées et non pas des moments.

La table *Durees* contient la colonne *dureeAnneesMois* (*INTERVAL YEAR TO MONTH*) pour stocker des intervalles en années et en jours, et la colonne *dureeJourSecondes* (*INTERVAL DAY TO SECOND*) pour stocker des intervalles en jours, heures, minutes, secondes et fractions de secondes.

```
CREATE TABLE Durees
  (dureeAnneesMois INTERVAL YEAR TO MONTH,
   dureeJourSecondes INTERVAL DAY TO SECOND);
```

L'insertion suivante initialise :

- la colonne *dureeAnneesMois* à la valeur d'1 an et 7 mois ;
- la colonne *dureeJourSecondes* à la valeur de 5 jours, 15 heures, 13 minutes, 56 secondes et 97 centièmes.

```
INSERT INTO Durees (dureeAnneesMois, dureeJourSecondes)
  VALUES ('1-7', '5 15:13:56.97');
```

Nous verrons comment ajouter ou soustraire un intervalle à une date ou à un autre intervalle.

Variables utiles



Les variables suivantes permettent de retrouver le moment de la session et le fuseau du serveur (si tant est qu'il soit déporté par rapport au client).

- *CURRENT_DATE* : date et heure de la session (format *DATE*) ;
- *LOCALTIMESTAMP* : date et heure de la session (format *TIMESTAMP*) ;
- *SYSTIMESTAMP* : date et heure du serveur (format *TIMESTAMP WITH TIME ZONE*) ;
- *DBTIMEZONE* : fuseau horaire du serveur (format *VARCHAR2*) ;
- *SESSIONTIMEZONE* : fuseau horaire de la session client (format *VARCHAR2*).

Il faut utiliser la pseudo-table *DUAL*, que nous détaillerons au chapitre 4, qui permet d'afficher une expression dans l'interface *SQL*Plus*.

L'exemple suivant montre que le script a été exécuté le 23 avril 2003 à 19 h 33, 8 secondes et 729 centièmes. Le client est sur le fuseau *GMT+2h*, le serveur quelque part aux États-Unis (*GMT-7*), option par défaut à l'installation d'Oracle. Ce dernier sait pertinemment qu'on a choisi la langue française mais a quand même laissé sa situation géographique. Il faudra la

modifier (dans le fichier de configuration) si on désire positionner le fuseau du serveur dans le même fuseau que le client.

```

SELECT CURRENT_DATE, LOCALTIMESTAMP, SYSTIMESTAMP, DBTIMEZONE,
SESSIONTIMEZONE FROM DUAL;
CURRENT_DATE      LOCALTIMESTAMP                SYSTIMESTAMP
-----
23/04/03          23/04/03 19:33:08,729000          23/04/03 19:33:08,729000 +02:00
DBTIMEZONE
-----
-07:00
                                +02:00
    
```

Caractères Unicode

Si vous envisagez de stocker des données qui ne sont ni des lettres, ni des chiffres, ni les symboles courants : *espace, tabulation, % ` ' () * + - , . / \ : ; < > = ! _ & ~ { } | ^ ? \$ # @ " []*, vous devrez utiliser, pour manipuler des caractères Unicode, les types NCHAR, NCHAR2 et NCLOB.

Le jeu de caractères d'Oracle pour une installation française est WE8ISO8859P1 (condensé de *Western Europe 8-bit ISO 8859 Part 1*). Le jeu de caractères national utilisé par défaut pour les types NCHAR est AL16UTF16.

Pour plus de détails, consultez le livre consacré au support du multilingue (*Database Globalization Support Guide*). Vous y découvrirez que, comme pour les caractères accentués (voir l'introduction), c'est au niveau du client (SQL Developer, Toad, SQL*Plus...) que vous devrez paramétrer des variables d'environnement ou des fichiers de configuration afin de pouvoir visualiser les données Unicode stockées. Par ailleurs, il est recommandé de préférer l'interface SQL Developer (plus apte à gérer des informations UTF-8) à SQL*Plus qui n'est pas adaptée et dédiée aux caractères ANSI ou ASCII. La fonction UNISTR sert à transformer un paramètre Unicode pour retourner une information codée dans le jeu de caractères de la base (AL16UTF16 ou UTF8).

Figure 2-2 Insertion de caractères Unicode

The screenshot shows a SQL query editor window with the following SQL code:

```
CREATE TABLE CaracteresUnicode
(col1 NVARCHAR2(10), col2 NVARCHAR2(10));
INSERT INTO CaracteresUnicode (col1,col2)
VALUES (N'copyright', UNISTR('\00A9'));
INSERT INTO CaracteresUnicode (col1,col2)
VALUES (N'du hindi', UNISTR('\0930\0941\092A\092F\093E'));
SELECT * FROM CaracteresUnicode ;
```

Below the code, the execution results are displayed:

```
table CARACTERESUNICODE créé(e).
1 lignes inséré.
1 lignes inséré.
COL1          COL2
-----
copyright     ©
du hindi      रुपया
```

Données LOB

Les types LOB (*Large Object Binary*) d'Oracle sont BLOB, CLOB, NCLOB et BFILE. Ils servent à stocker de grandes quantités de données non structurées (textes, images, vidéos, sons). Ils succèdent aux types LONG. Les LOB sont étudiés plus en détail dans la partie consacrée à la programmation PL/SQL.

Considérons la table suivante.

```
CREATE TABLE Trombinoscope (nomEtudiant VARCHAR(30), photo BFILE);
```

Le stockage de l'image photoCS.jpg, qui se trouve à l'extérieur de la base (dans le répertoire D:\PhotosEtudiant), est réalisé par l'insertion dans la colonne BFILE d'un pointeur (*locator*) qui adresse le fichier externe via la fonction BFILENAME du paquetage DBMS_LOB. L'utilisateur doit avoir reçu au préalable le privilège CREATE ANY DIRECTORY.

```
CREATE DIRECTORY repertoire_etudiants AS 'D:\PhotosEtudiant';
INSERT INTO Trombinoscope
VALUES ('Soutou', BFILENAME('repertoire_etudiants', 'photoCS.jpg'));
```

L'interface en mode texte SQL*Plus n'est pas capable d'afficher cette image. Il faudra pour cela utiliser un logiciel approprié (une interface Web ou Java par exemple, après avoir chargé cette image par la fonction LOADFROMFILE du paquetage DBMS_LOB).

Séquences

Une séquence est un objet de schéma (appartenant à l'utilisateur qui l'a créé) qui a pour objectif de générer automatiquement des valeurs (de type NUMBER). Bien qu'elles soient majoritairement utilisées pour composer des valeurs auto-incrémentées pour les clés primaires, il est possible de les employer au sein de différentes tables.

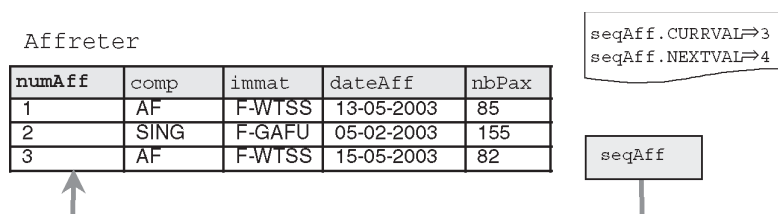
Gérée indépendamment d'une table, une séquence peut être partagée par plusieurs utilisateurs.



Depuis la version 12c, il est possible de définir une colonne auto-incrémentée à l'aide de la directive `GENERATED... AS IDENTITY...` avec les mêmes options dédiées initialement aux séquences (voir l'instruction `CREATE SEQUENCE`). La directive d'auto-incrémentation peut être utilisée dans une instruction `CREATE TABLE` ou `ALTER TABLE`.

La figure suivante illustre la séquence `seqAff` utilisée pour initialiser les valeurs de la clé primaire `numAff` de la table `Affreter`. Seules deux fonctions (aussi appelées pseudo-colonnes ou directives) peuvent être appliquées à une séquence : `CURRVAL` retourne la valeur courante, `NEXTVAL` incrémente la séquence et retourne la valeur obtenue (ici le pas est de 1, nous verrons qu'il peut être différent de cette valeur).

Figure 2-3 Séquence appliquée à une clé primaire



Création d'une séquence (CREATE SEQUENCE)

Vous devez avoir le privilège `CREATE SEQUENCE` pour pouvoir créer une séquence dans votre schéma. Pour en créer une dans un schéma différent du vôtre, le privilège `CREATE ANY SEQUENCE` est requis.

La syntaxe de création d'une séquence est la suivante :

```

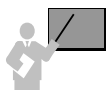
CREATE SEQUENCE [schéma.]nomSéquence
  [INCREMENT BY entier ]
  [START WITH entier ]
  [ { MAXVALUE entier | NOMAXVALUE } ]
  [ { MINVALUE entier | NOMINVALUE } ]
  [ { CYCLE | NOCYCLE } ]
  [ { CACHE entier | NOCACHE } ]
  [ { ORDER | NOORDER } ] ;

```

Si aucun nom de schéma n'est spécifié la séquence créée vous appartient. Si aucune option n'est précisée, la séquence créée commencera à 1 et augmentera sans fin (la limite réelle d'une séquence est de $10^{29}-1$). En spécifiant seulement « INCREMENT BY -1 » la séquence créée commencera à -1 et sa valeur diminuera sans limites (la borne inférieure réelle d'une séquence est de $-10^{27}-1$).

- **INCREMENT BY** : donne l'intervalle entre deux valeurs de la séquence (entier positif ou négatif mais pas nul). La valeur absolue de cet intervalle doit être plus petite que ($MAXVALUE-MINVALUE$). L'intervalle par défaut est 1.
- **START WITH** : précise la première valeur de la séquence à générer. Pour les séquences ascendantes (d'un incrément positif), la valeur par défaut est égale à la valeur minimale de la séquence. Pour les séquences descendantes, la valeur par défaut est égale à la valeur maximale de la séquence (entier jusqu'à $10^{28}-1$, pour les négatifs : $-10^{27}+1$).
- **MAXVALUE** : donne la valeur maximale de la séquence. Cette limite doit être supérieure ou égale à l'entier défini dans **START WITH** et supérieure à **MINVALUE**.
- **NOMAXVALUE** (par défaut) fixe le maximum à $10^{28}-1$ pour une séquence ascendante et à $-10^{27}+1$ pour une séquence descendante.
- **MINVALUE** précise la valeur minimale de la séquence. Cette limite doit être inférieure ou égale à l'entier défini dans **START WITH** et inférieure à **MAXVALUE**.
- **NOMINVALUE** (par défaut) fixe le minimum à 1 pour une séquence ascendante et à la valeur $-10^{27}-1$ pour une séquence descendante.
- **CYCLE** indique que la séquence doit continuer de générer des valeurs même après avoir atteint sa limite. Au-delà de la valeur maximale, la séquence générera la valeur minimale et incrémentera comme cela est défini dans la clause concernée. Après la valeur minimale, la séquence produira la valeur maximale et décrémentera comme cela est défini dans la clause concernée.
- **NOCYCLE** (par défaut) indique que la séquence ne doit plus générer de valeurs une fois la limite atteinte.
- **CACHE** fixe le nombre de valeurs de la séquence que le cache va contenir (et qui évite la sollicitation du compteur en temps réel). Le minimum est 2 et le maximum théorique est fonction d'une formule analogue à $maxi_séquence-mini_séquence/increment_séquence$ (par exemple, pour une séquence de valeur maximale 50 000 et de valeur minimale 1 avec

un pas de 2, le nombre maximal de valeurs en cache serait de 25 000). Par défaut, le cache contient 20 valeurs.



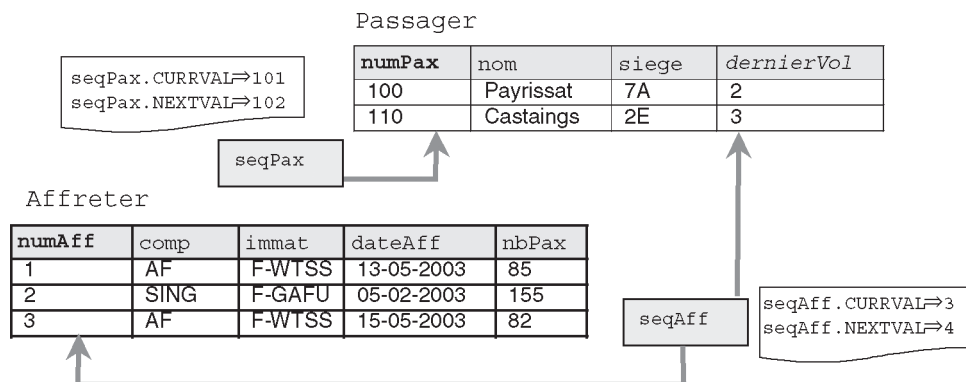
En fonction du nombre de séquences mises en cache, et selon l'utilisation (même si d'autres utilisateurs ou d'autres connexions emploient cette séquence) voire l'interruption du serveur, il est possible que des valeurs retournées ne se succèdent pas d'une seule valeur du pas de l'incrément du fait du cache perdu. En revanche, le contrat qu'Oracle remplit consiste à délivrer toujours une nouvelle valeur après l'appel de NEXTVAL.

- ORDER garantit que les valeurs de la séquence sont générées dans l'ordre des requêtes. Si vos séquences jouent le rôle d'horodatage (*timestamp*), vous devrez utiliser cette option. Pour la génération de clés primaires, cette option n'est pas importante.

Créons les deux séquences (*seqAff* et *seqPax*) qui vont permettre de donner leur valeur aux clés primaires des deux tables illustrées à la figure suivante. On suppose qu'on ne stockera pas plus de 100 000 passagers et pas plus de 10 000 affrètements.

Servons-nous aussi de la séquence *seqAff* dans la table *Passager* pour indiquer le dernier vol de chaque passager. *seqAff* sert à donner leur valeur à la clé primaire de *Affreter* et à la clé étrangère de *Passager*. La section « Intégrité référentielle » détaille les mécanismes relatifs aux clés étrangères.

Figure 2-4 Séquences



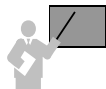
Le script SQL de définition des données est indiqué ci-après. Notez que les déclarations sont indépendantes, ce n'est qu'au moment des insertions qu'on affectera aux colonnes concernées les valeurs des séquences.

Tableau 2-3 Tables et séquences

Tables	Séquences
<pre>CREATE TABLE Affreter (numAff NUMBER(5), comp VARCHAR2(4), immat VARCHAR2(6), dateAff DATE, nbPax NUMBER(3), CONSTRAINT pk_Affreter PRIMARY KEY (numAff)); CREATE TABLE Passager (numPax NUMBER(6), nom VARCHAR2(15), siege VARCHAR2(4), dernierVol NUMBER(5), CONSTRAINT pk_Passager PRIMARY KEY (numPax), CONSTRAINT fk_Pax_vol_Affreter FOREIGN KEY (dernierVol) REFERENCES Affreter (numAff));</pre>	<pre>CREATE SEQUENCE seqAff MAXVALUE 10000 NOMINVALUE;</pre> <pre>CREATE SEQUENCE seqPax INCREMENT BY 10 START WITH 100 MAXVALUE 100000 NOMINVALUE;</pre>

Manipulation d'une séquence

Vous devez avoir le privilège `SELECT` sur une séquence (privilège donné par `GRANT SELECT ON seq TO utilisateur`) pour pouvoir en utiliser une. Pour manipuler une séquence dans un schéma différent du vôtre, le privilège `SELECT ANY SEQUENCE` est requis. Dans ce cas il faudra toujours préfixer le nom de la séquence par celui du schéma (par exemple *jean.seq*).



Une fois créée, une séquence *seq* ne peut se manipuler que via deux directives (qu'Oracle appelle aussi pseudo-colonnes) :

- *seq.CURRVAL* qui retourne la valeur courante de la séquence (lecture seule) ;
- *seq.NEXTVAL* qui incrémente la séquence et retourne la nouvelle valeur de celle-ci (écriture et lecture).

Le premier appel à `NEXTVAL` retourne la valeur initiale de la séquence (définie dans `START WITH`). Les appels suivants augmentent la séquence de la valeur définie dans `INCREMENT WITH`.

Chaque appel à `CURRVAL` retourne la valeur courante de la séquence. Il faut utiliser au moins une fois `NEXTVAL` avant d'appeler `CURRVAL` dans une même session (SQL*Plus, bloc PL/SQL ou programme). Ces directives peuvent s'utiliser :

- au premier niveau d'une requête `SELECT` (voir le chapitre 4) ;
- dans la clause `SELECT` d'une instruction `INSERT` (voir la section « Insertion multilignes » du chapitre 4) ;
- dans la clause `VALUES` d'une instruction `INSERT` (voir l'exemple suivant) ;
- dans la clause `SET` d'une instruction `UPDATE` (voir la section ci-après).



Les principales restrictions d'utilisation de NEXTVAL et CURRVAL sont :

- sous-interrogation dans une instruction DELETE, SELECT, ou UPDATE (voir le chapitre 4) ;
- dans un SELECT d'une vue (voir le chapitre 5) ;
- dans un SELECT utilisant DISTINCT, GROUP BY, ORDER BY ou des opérateurs ensemblistes (voir le chapitre 4) ;
- en tant que valeur par défaut (DEFAULT) d'une colonne d'un CREATE TABLE ou ALTER TABLE ;
- dans la condition d'une contrainte CHECK d'un CREATE TABLE ou ALTER TABLE.

Le tableau suivant illustre l'évolution de nos deux séquences en fonction de l'insertion des enregistrements décrits dans la figure précédente. Nous utilisons NEXTVAL pour les clés primaires et CURRVAL pour la clé étrangère (de manière à récupérer la dernière valeur de la séquence utilisée pour la clé primaire).

Tableau 2-4 Manipulation de séquences

Instructions SQL	Séquences	seqAff		seqPax	
		CURRVAL	NEXTVAL	CURRVAL	NEXTVAL
--Aucune insertion encore		Pas définies			
INSERT INTO Affreter VALUES (seqAff.NEXTVAL, 'AF', 'F-WTSS', '13-05-2003', 85);		1	2	Pas définies	
INSERT INTO Affreter VALUES (seqAff.NEXTVAL, 'SING', 'F-GAFU', '05-02-2003', 155);					
INSERT INTO Passager VALUES (seqPax.NEXTVAL, 'Payrissat', '7A', seqAff.CURRVAL);		2	3		
INSERT INTO Affreter VALUES (seqAff.NEXTVAL, 'AF', 'F-WTSS', '15-05-2003', 82);				100	110
INSERT INTO Passager VALUES (seqPax.NEXTVAL, 'Castaings', '2E', seqAff.CURRVAL);		3	4	110	120

Utilisation d'une séquence dans un DEFAULT



Depuis la version 12c, la valeur par défaut d'une colonne numérique entière peut être définie avec NEXTVAL ou CURRVAL. Ainsi, lors d'insertions, la génération automatique de valeurs se produira en utilisant la séquence précisée dans la clause DEFAULT.

Le code suivant présente l'utilisation de cette option pour deux séquences dédiées à une clé primaire et à une clé étrangère, sous réserve que les transactions s'exécutent toujours dans l'ordre : ajout du vol, puis des passagers. La génération des clés étrangères ne perturbe pas la valeur de la clé primaire qui est gérée par la séquence principale.

Tableau 2-5 Séquences par défaut sur une colonne

Création des séquences et mise en place	Insertions
<pre>CREATE SEQUENCE master_seq; CREATE SEQUENCE detail_seq; CREATE TABLE vol (id NUMBER DEFAULT master_seq. NEXTVAL, description VARCHAR2(6), jour_vol DATE); CREATE TABLE places (id NUMBER DEFAULT detail_seq.NEXTVAL, vol_id NUMBER DEFAULT master_seq. CURRVAL, pax_nom VARCHAR2(30));</pre>	<pre>INSERT INTO vol (description, jour_vol) VALUES ('AF6143', SYSDATE-1); INSERT INTO places (pax_nom) VALUES ('Joppé'); INSERT INTO places (pax_nom) VALUES ('Rienna'); INSERT INTO places (pax_nom) VALUES ('Guilbaud');</pre>

Figure 2-5 Séquences par défaut

VOL			PLACES		
ID	DESCRIPTION	JOUR_VOL	ID	VOL_ID	PAX_NOM
1	AF6143	19/11/14	1	1	Joppé
			2	1	Rienna
			3	1	Guilbaud

Modification d'une séquence (ALTER SEQUENCE)

Vous devez avoir le privilège ALTER SEQUENCE pour pouvoir modifier une séquence de votre schéma. Pour modifier une séquence dans un schéma différent du vôtre, le privilège ALTER ANY SEQUENCE est requis.

Les modifications les plus courantes sont celles qui consistent à augmenter les limites d'une séquence ou à changer le pas de son incrémentation. Dans tous les cas, seules les valeurs à venir de la séquence modifiée seront changées (heureusement pour les données existantes des tables).

La syntaxe de modification d'une séquence reprend la plupart des éléments de sa création.

```
ALTER SEQUENCE [schéma.]nomSéquence
  [INCREMENT BY entier ]
  [ { MAXVALUE entier | NOMAXVALUE } ]
  [ { MINVALUE entier | NOMINVALUE } ]
  [ { CYCLE | NOCYCLE } ]
  [ { CACHE entier | NOCACHE } ]
  [ { ORDER | NOORDER } ] ;
```

La clause `START WITH` ne peut être modifiée sans supprimer et recréer la séquence. Des contrôles sont opérés sur les limites, par exemple `MAXVALUE` ne peut pas être affectée à une valeur plus petite que la valeur courante de la séquence.

Supposons qu'on ne stockera pas plus de 95 000 passagers et pas plus de 850 affrètements. De plus les incréments des séquences doivent être égaux à 5. Les instructions SQL à appliquer sont les suivantes : chaque invocation des méthodes `NEXTVAL` prendra en compte désormais le nouvel incrément tout en laissant intactes les données existantes des tables.

```
ALTER SEQUENCE seqAff INCREMENT BY 5 MAXVALUE 850;
ALTER SEQUENCE seqPax INCREMENT BY 5 MAXVALUE 95000;
```

Visualisation d'une séquence

La pseudo-table `DUAL` peut être utilisée pour visualiser le contenu d'une séquence. En appliquant la directive `CURRVAL` on extrait le contenu actuel de la séquence (la dernière valeur générée).



En appliquant la directive `NEXTVAL` dans un `SELECT` la séquence s'incrémente avant de s'afficher. Vous réalisez alors un effet de bord car la valeur qui apparaît à l'écran est désormais perdue pour une éventuelle utilisation dans une clé primaire.

Le tableau suivant illustre l'utilisation de la pseudo-table `DUAL` pour visualiser les séquences créées auparavant.

Tableau 2-6 Visualisation de séquences

Besoin	Requête SQL et résultat sous SQL*Plus
Quelles sont les dernières valeurs générées par mes séquences ?	<pre>SELECT seqAff.CURRVAL "seqAff (CURRVAL)" , seqPax.CURRVAL "seqPax (CURRVAL)" FROM DUAL; seqAff (CURRVAL) seqPax (CURRVAL) ----- 3 110</pre>
Quelles sont les prochaines valeurs produites par mes séquences ? (qui sont perdues car les incréments s'opèrent lors de la requête)	<pre>SELECT seqAff.NEXTVAL "seqAff (NEXTVAL)" , seqPax.NEXTVAL "seqPax (NEXTVAL)" FROM DUAL; seqAff (NEXTVAL) seqPax (NEXTVAL) ----- 4 120</pre>

Suppression d'une séquence (DROP SEQUENCE)

L'instruction `DROP SEQUENCE` supprime une séquence. Celle-ci doit se trouver dans votre schéma (vous en êtes propriétaire) ou vous devez avoir le privilège `DROP ANY SEQUENCE`.

La suppression d'une séquence peut être utilisée pour refaire partir une séquence donnée à un chiffre nouveau (clause `START WITH`). En ce cas, il faut bien sûr recréer la séquence après l'avoir supprimée.

La syntaxe de suppression d'une séquence est la suivante.

```
DROP SEQUENCE [schéma.]nomSéquence ;
```

Supprimons les deux séquences de notre schéma par les instructions suivantes :

```
DROP SEQUENCE seqAff;
DROP SEQUENCE seqPax;
```

Colonnes auto-incrémentées



Depuis la version 12c, il est possible d'utiliser un type numérique (entier) pour définir une colonne auto-incrémentée avec la clause `GENERATED... AS IDENTITY...` disponible dans les instructions `CREATE TABLE` et `ALTER TABLE`.

Avant de détailler les options d'auto-incrémentation, vous devez savoir qu'une colonne auto-incrémentée dispose en interne d'une séquence générée automatiquement et qui lui est dédiée. Ainsi, vous retrouverez les options `INCREMENT`, `START...` lorsque vous définirez un auto-incrément.

```
GENERATED [ALWAYS | BY DEFAULT [ON NULL]]
AS IDENTITY [(options_séquence)]
```

- ALWAYS (par défaut) utilise le générateur de séquences et interdit qu'une valeur soit explicitement imposée lors d'un INSERT ou UPDATE (erreur ORA-32795 impossible d'insérer la valeur dans une colonne d'identité..).
- BY DEFAULT utilise le générateur de séquences mais n'interdit pas qu'une valeur soit explicitement imposée d'un INSERT ou UPDATE. Avec l'option ON NULL, la séquence est capable d'affecter implicitement une valeur à chaque INSERT ou si un quelconque NULL arrive en lieu et place de la colonne concernée.
- *options_sequence* sont identiques à celles du CREATE SEQUENCE.

Le code suivant présente l'utilisation de cette option pour une clé primaire. Il est à noter que le NOT NULL est implicite sur une telle colonne et que vous ne pouvez disposer que d'un seul auto-incrément par table.

Tableau 2-7 Colonne auto-incrémentée

Création de la table et de son auto-incrémentation	Insertions
<pre>CREATE TABLE billets (id NUMBER(6) GENERATED ALWAYS AS IDENTITY, vol_id VARCHAR2(6) NOT NULL, jour_vol DATE DEFAULT SYSDATE NOT NULL, pax_nom VARCHAR2(30) NOT NULL, siege_pax CHAR(3) NOT NULL , CONSTRAINT pk_billets PRIMARY KEY(id));</pre>	<pre>INSERT INTO billets(vol_id,pax_nom,siege_pax) VALUES ('AF6143', 'Guilbaud', '03F'); INSERT INTO billets(vol_id,pax_nom,siege_pax) VALUES ('AF6145', 'Blanchet', '23B'); INSERT INTO billets(vol_id,pax_nom,siege_pax) VALUES ('AF6145', 'Bruchez', '02A');</pre>

Figure 2-6 Colonne auto-incrémentée

SQL> SELECT * FROM billets;

ID	UOL_ID	JOUR_UOL	PAX_NOM	SIEGE_PAX
1	AF6143	20/11/14	Guilbaud	03F
2	AF6145	20/11/14	Blanchet	23B
3	AF6145	20/11/14	Bruchez	02A

Modifications de valeurs

L'instruction UPDATE permet la mise à jour des colonnes d'une table. Pour pouvoir modifier des enregistrements d'une table, il faut que cette dernière soit dans votre schéma ou que vous ayez reçu le privilège UPDATE sur la table. Si vous avez le privilège UPDATE ANY TABLE, vous pouvez modifier des enregistrements de tout schéma.

Syntaxe (UPDATE)

La syntaxe simplifiée de l'instruction UPDATE est la suivante.

```
UPDATE [schéma.] nomTable
SET  colonne1 = { expression | (requête_SELECT) | DEFAULT }
     [colonne2 ...]
```

La première écriture de la clause SET met à jour une colonne en lui affectant une expression (valeur, valeur par défaut, calcul, résultat d'une requête). La deuxième écriture rafraîchit plusieurs colonnes à l'aide du résultat d'une requête.

La condition filtre les lignes à mettre à jour dans la table. Si aucune condition n'est précisée, tous les enregistrements seront mis à jour. Si la condition ne filtre aucune ligne, aucune mise à jour ne sera réalisée.

Modification d'une ligne

Affectons un nom à la compagnie de code 'GO' et modifions sa date de création.

```
UPDATE compagnie
SET    nom_comp      = 'Go Airways',
        date_creation = TO_DATE('30/12/2014', 'DD/MM/YYYY')
WHERE  comp = 'GO';
```

Modification de plusieurs lignes

Pour remplacer le marqueur NULL par le nombre 10, il suffit de conditionner la mise à jour à l'aide de la fonction IS NULL (voir le chapitre 4). Attention, si vous utilisez la condition WHERE nb_passagers = NULL, vous ne sélectionnerez aucune ligne car deux NULL sont différents entre eux.

```
UPDATE vol_jour
SET    nb_passagers = 10
WHERE  nb_passagers IS NULL;
```

Les modifications sont présentées dans la figure suivante.

Figure 2-7 Table après les modifications

COMP	NOM_COMP	DATE_CREATION
SING	Singapore AL	01/01/47
AF	Air France	01/01/33
GO	(null)	31/12/14

NUM_VOL	AERO_DEP	AERO_ARR	COMP	JOUR_VOL	NB_PASSAGERS
AF6143	TLS	ORY	AF	20/11/14	120
AF6550	CDG	TLS	AF	20/11/14	195
SQ747	CDG	SIN	SING	20/11/14	(null)
AF6145	ORY	TLS	AF	20/11/14	(null)

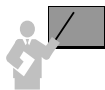
Ne pas respecter des contraintes

Il faut, comme pour les insertions, respecter les contraintes qui existent au niveau des colonnes. Dans le cas inverse, une erreur est renvoyée (le nom de la contrainte apparaît) et la mise à jour n'est pas effectuée.

Le tableau suivant décrit une tentative de modification pour chaque type de contrainte que vous pourrez être amené à rencontrer. Les problèmes présentés ici sont respectivement une clé primaire en doublon, une colonne obligatoire, un aéroport de départ semblable à celui d'arrivée, un libellé dupliqué et une compagnie inexistante.

Tableau 2-8 Modifications impossibles

Type de contrainte	Instructions SQL et résultats
Clé primaire	SQL> UPDATE compagnie SET comp = 'AF' WHERE comp = 'GO'; ERREUR : ORA-00001: violation de contrainte unique (SOUTOU.PK_COMPAGNIE)
Non-nullité	SQL> UPDATE vol_jour SET aero_dep = NULL WHERE num_vol = 'AF6143'; ERREUR : ORA-01407: impossible de mettre à jour ("SOUTOU"."VOL_JOUR"."AERO_DEP") avec NULL
Vérification	SQL> UPDATE vol_jour SET aero_arr = 'TLS' WHERE num_vol = 'AF6143'; ERREUR : ORA-02290: violation de contraintes (SOUTOU.CK_TRAJET) de vérification
Unicité	SQL> UPDATE compagnie SET nom_comp = 'Go Airways' WHERE comp = 'AF'; ERREUR : ORA-00001: violation de contrainte unique (SOUTOU.UN_NOM_COMP)
Clé étrangère	SQL> UPDATE vol_jour SET comp = 'EJET' WHERE num_vol = 'AF6143'; ERREUR : ORA-02291: violation de contrainte d'intégrité (SOUTOU.FK_VOL_JOUR_COMP_COMPAGNIE) - clé parent introuvable



La mise à jour d'une clé étrangère est possible si la nouvelle valeur est bien référencée. La mise à jour d'une clé primaire est possible si aucune ligne d'aucune table ne la référence déjà (voir la section « Intégrité référentielle »).

Dates et intervalles

Le tableau suivant résume les opérations possibles entre des colonnes de type DATE et Interval.

Tableau 2-9 Opérations entre dates et intervalles

Opérande 1	Opérateur	Opérande 2	Résultat
DATE	+ ou -	INTERVAL	DATE
DATE	+ ou -	NUMBER	DATE
Interval	+	DATE	DATE
DATE	-	DATE	NUMBER
Interval	+ ou -	INTERVAL	INTERVAL
Interval	* ou /	NUMBER	INTERVAL

Considérons la table suivante :

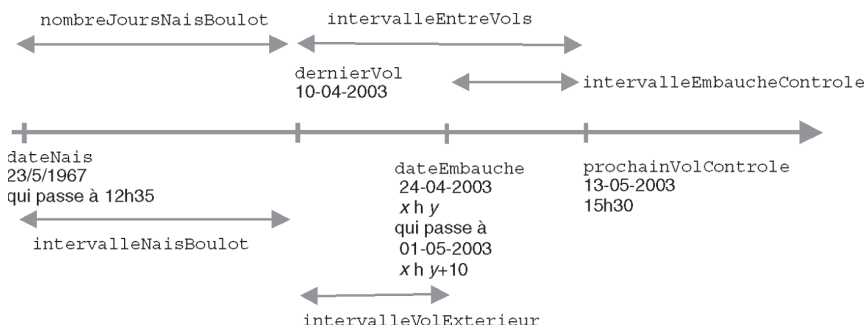
```
CREATE TABLE Pilote
(brevet VARCHAR(6), nom VARCHAR(20), dateNaiss DATE, dernierVol DATE,
dateEmbauche DATE, prochainVolControle DATE,
nombreJoursNaisBoulot NUMBER,
intervalleNaisBoulot INTERVAL DAY(7) TO SECOND(3),
intervalleVolExterieur INTERVAL DAY(2) TO SECOND(0),
intervalleEntreVols INTERVAL DAY(2) TO SECOND(2),
intervalleEmbaucheControle INTERVAL DAY(2) TO SECOND(1),
compa VARCHAR(4), CONSTRAINT pk_Pilote PRIMARY KEY(brevet));
```

À l'insertion du pilote, nous initialisons sa date de naissance, la date de son dernier vol, sa date d'embauche (à celle du jour via SYSDATE) et la date de son prochain contrôle en vol au 13 mai 2003, 15 h 30 (heures et minutes évaluées à l'aide de la fonction TO_DATE qui convertit une chaîne en date).

```
INSERT INTO Pilote
VALUES ('PL-1', 'Thierry Albaric', '25-03-1967', '10-04-2003', SYSDATE,
TO_DATE('13-05-2003 15:30:00', 'DD:MM:YYYY HH24:MI:SS'), NULL, NULL,
NULL, NULL, NULL, 'AF');
```

Les mises à jour par UPDATE sur cet enregistrement vont consister, sur la base de ces quatre dates, à calculer les intervalles illustrés à la figure suivante :

Figure 2-8 Intervalles à calculer



Modification d'une heure

On modifie une date en précisant une heure via la fonction TO_DATE.

```
UPDATE Pilote
SET dateNaiss = TO_DATE('25-03-1967 12:35:00',
'DD:MM:YYYY HH24:MI:SS')
WHERE brevet = 'PL-1';
```

Ajout d'un délai

On modifie la date d'embauche de 10 minutes après la semaine prochaine. L'ajout d'une semaine se fait par l'opération +7 à une date. L'addition de 10 minutes se fait par l'ajout de la fraction de jour correspondante (10/(24*60)).

```
UPDATE Pilote
SET dateEmbauche = dateEmbauche + 7 + (10/(24*60)) WHERE brevet =
'PL-1';
```

Différence entre deux dates

La différence entre deux dates renvoie un entier correspondant au nombre de jours.

```
UPDATE Pilote
SET nombreJoursNaisBoulot = dateEmbauche-dateNaiss WHERE brevet =
'PL-1';
```

Cette même différence au format INTERVAL en nombre de jours requiert l'utilisation de la fonction NUMTODSINTERVAL.

```
UPDATE Pilote
SET intervalleNaisBoulot =
NUMTODSINTERVAL(dateEmbauche-dateNaiss, 'DAY'),
```

```

intervalleEntreVols =
NUMTODSINTERVAL(prochainVolControle-dernierVol, 'DAY'),
intervalleVolExterieur =
NUMTODSINTERVAL(dateEmbauche-dernierVol, 'DAY')
WHERE brevet = 'PL-1';

```

Différence entre deux intervalles

La différence entre deux intervalles homogènes renvoie un intervalle.

```

UPDATE Pilote
SET intervalleEmbaucheControle =
intervalleEntreVols-intervalleVolExterieur
WHERE brevet = 'PL-1';

```

La ligne contient désormais les informations suivantes. Les données en gras correspondent aux mises à jour. On trouve qu'il a fallu 13 186 jours, 3 heures, 49 minutes et 53 secondes pour que ce pilote soit embauché. 21 jours, 16 heures, 24 minutes et 53 secondes séparent le dernier vol du pilote au moment de son embauche. 33 jours, 15 heures et 30 minutes séparent son dernier vol de son prochain contrôle en vol. La différence entre ces deux délais est de 11 jours, 23 heures, 5 minutes et 7 secondes.

Figure 2-9 Ligne modifiée par des calculs de dates

Pilote

brevet	nom	dateNaiss	dernierVol	dateEmbauche	prochainVolControle
PL-1	Thierry Albaric	25-03-1967 25-03-1967 12:35:00	10-04-2003	24:04:2003 04:14:53 01:05:2003 04:24:53	13-05-2003 15:30:00
nombreJoursNaisBoulot	intervalleNaisBoulot	intervalleVolExterieur			
13186,1596	+0013186 03:49:53.000	+21 16:24:53			
intervalleEntreVols	intervalleEmbaucheControle	compa			
+33 15:30:00.00	+11 23:05:07.0	AF			

Fonctions utiles



Les fonctions suivantes vous seront d'un grand secours pour manipuler des dates et des intervalles.

- `TO_CHAR(colonneDate [, format [, 'NLS_DATE_LANGUAGE=Langue']])` convertit une date en chaîne suivant un certain format dans un certain langage ;
- `TO_DATE(chaineCaractères [, format [, 'NLS_DATE_LANGUAGE=Langue']])` convertit une chaîne en date suivant un certain format dans un certain langage ;

- `EXTRACT({YEAR | MONTH | DAY | HOUR | MINUTE | SECOND} FROM {expression-DATE | expressionINTERVAL})` extrait une partie donnée d'une date ou d'un intervalle ;
- `NUMTOYMINTERVAL(expressionNumérique, {'YEAR' | 'MONTH'})` convertit un nombre dans un type `INTERVAL YEAR TO MONTH` ;
- `NUMTODSINTERVAL(expressionNumérique, {'DAY' | 'HOUR' | 'MINUTE' | 'SECOND'})` convertit un nombre dans un type `INTERVAL DAY TO SECOND`.

Les tableaux suivants présentent quelques exemples d'utilisation de ces fonctions.

Tableau 2-10 Quelques formats pour TO_CHAR

Expression	Résultats	Commentaires
<code>TO_CHAR(dateNaiss , 'J')</code>	2439575	Le calendrier julien est utilisé ici (comptage du nombre de jours depuis le 1 ^{er} janvier, 4712 av. J.-C. jusqu'au 25 mars 1967).
<code>TO_CHAR(dateNaiss, 'DAY - MONTH - YEAR')</code>	SAMEDI - MARS - NINETEEN SIXTY-SEVEN	Affichage des libellés des jours, mois et années. Oracle ne traduit pas encore notre année.
<code>TO_CHAR(dateEmbauche , 'DDD')</code>	121	Affichage du numéro du jour de l'année (ici il s'agit 1 ^{er} mai 2003).

Tableau 2-11 Quelques formats pour TO_DATE

Expression	Commentaires
<code>TO_DATE('May 13, 1995, 12:30 A.M.', 'MONTH DD, YYYY, HH:MI A.M.', 'NLS_DATE_LANGUAGE = American')</code>	Définition d'une date à partir d'un libellé au format américain.
<code>TO_DATE('13 Mai, 1995, 12:30', 'DD MONTH, YYYY, HH24:MI', 'NLS_DATE_LANGUAGE = French')</code>	Définition de la même date pour les francophones (l'option NLS par défaut est la langue à l'installation).

Tableau 2-12 Utilisation de EXTRACT

Expression	Résultats	Commentaires
<code>EXTRACT(DAY FROM intervalleVolExterieur)</code>	21	Extraction du nombre de jours dans l'intervalle contenu dans la colonne.
<code>EXTRACT(MONTH FROM dateNaiss)</code>	3	Extraction du mois de la date contenue dans la colonne.

Tableau 2-13 Conversion en intervalles

Expression	Résultats	Commentaires
<code>NUMTOYMINTERVAL(1.54, 'YEAR')</code>	+000000001-06	1 an et 54 centièmes d'année est converti en 1 an et 6 mois.
<code>NUMTOYMINTERVAL(1.54, 'MONTH')</code>	+000000000-01	1 mois et 54 centièmes de mois est converti en 1 mois à l'arrondi.
<code>NUMTODSINTERVAL(1.54, 'DAY')</code>	+000000001 12:57:36.00	1 jour et 54 centièmes de jour est converti en 1 jour, 12 heures, 57 minutes et 36 secondes.
<code>NUMTODSINTERVAL(1.54, 'HOUR')</code>	+000000000 01:32:24.00	1 heure et 54 centièmes est converti en 1 heure, 32 minutes et 24 secondes.

Suppressions d'enregistrements

Les instructions `DELETE` et `TRUNCATE` permettent de supprimer un ou plusieurs enregistrements d'une table. Pour pouvoir supprimer des données dans une table, il faut que cette dernière soit dans votre schéma ou que vous ayez reçu le privilège `DELETE` sur la table. Si vous avez le privilège `DELETE ANY TABLE`, vous pouvez détruire des enregistrements dans n'importe quelle table de tout schéma.

Instruction `DELETE`

La syntaxe simplifiée de l'instruction `DELETE` est la suivante :

```
DELETE FROM [schéma.]nomTable [WHERE condition];
```

La condition sélectionne les lignes à supprimer. Si aucune condition n'est précisée, toutes les lignes sont supprimées. Si l'expression ne sélectionne aucune ligne, rien ne sera supprimé et aucune erreur n'est retournée. Supprimons une compagnie et un vol journalier (notez qu'il faut préciser l'heure dans le format de date si une heure a été incluse dans la date) à l'aide de cette instruction :

```
SQL> DELETE FROM compagnie WHERE nom_comp = 'Go Airways';
1 ligne supprimée.
SQL> DELETE FROM vol_jour
      WHERE jour_vol = TO_DATE('20/11/2014 15:30', 'DD/MM/YYYY HH24:MI')
      AND num_vol = 'AF6143';
1 ligne supprimée.
```

Les suppressions sont présentées dans la figure suivante.

Figure 2-10 Table après les suppressions

COMPAGNIE *		
COMP	NOM_COMP	DATE_CREATION
SING	Singapore AL	01/01/47
AF	Air France	01/01/33
GO	Go Airways	30/12/14

VOL_JOUR *					
NUM_VOL	AERO_DEP	AERO_ARR	COMP	JOUR_VOL	NB_PASSAGERS
AF6550	CDG	TLS	AF	20/11/14	195
AF6143	TLS	ORY	AF	20/11/14	120
SQ747	CDG	SIN	SING	20/11/14	10
AF6145	ORY	TLS	AF	20/11/14	10



La suppression d'une ligne contenant une clé étrangère est possible si cette même ligne ne joue pas le rôle de référent (cible d'une clé étrangère d'une autre table). La suppression d'une ligne pour laquelle la clé primaire est utilisée dans une autre table en tant que clé étrangère n'est pas possible sans un mécanisme de cascade (voir la section « Intégrité référentielle »).

Tentons de supprimer une compagnie qui est référencée par un pilote à l'aide d'une clé étrangère. Une erreur s'affiche, laquelle sera expliquée dans la section « Intégrité référentielle ».

```
DELETE FROM Compagnie WHERE comp = 'SING';
ORA-02292: violation de contrainte (SOUTOU.FK_PIL_COMPA_COMP) d'intégrité - enregistrement fils existant
```

Instruction TRUNCATE

La commande TRUNCATE supprime tous les enregistrements d'une table et libère éventuellement l'espace de stockage utilisé par la table (chose que ne peut pas faire DELETE) :

```
TRUNCATE TABLE [schéma.]nomTable [{ DROP | REUSE } STORAGE];
```



Il n'est pas possible de tronquer une table qui est référencée par des clés étrangères actives (sauf si la clé étrangère est elle-même dans la table à supprimer). La solution consiste à désactiver les contraintes puis à tronquer la table.

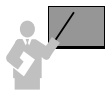
La récupération de l'espace est réalisée à l'aide de l'option DROP STORAGE (option par défaut). Dans le cas inverse (REUSE STORAGE), l'espace est utilisable par les nouvelles données de la table.

Intégrité référentielle

Les contraintes référentielles forment le cœur de la cohérence d'une base de données relationnelle. Ces contraintes sont fondées sur une relation entre clés étrangères et clés primaires et permettent de programmer des règles de gestion (exemple : l'affrètement d'un avion doit se faire par une compagnie existant dans la base de données). Ce faisant, les contrôles côté client (interface) sont ainsi déportés côté serveur.

C'est seulement dans sa version 7 en 1992, qu'Oracle a inclus dans son offre les contraintes référentielles.

Pour les règles de gestion trop complexes (exemple : l'affrètement d'un avion doit se faire par une compagnie qui a embauché au moins quinze pilotes dans les six derniers mois), il faudra programmer un déclencheur (voir le chapitre 7). Il faut savoir que les déclencheurs sont plus pénalisants que des contraintes dans un mode transactionnel (lectures consistantes).

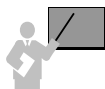


La contrainte référentielle concerne toujours deux tables – une table « père » aussi dite « maître » (*parent/referenced*) et une table « fils » (*child/dependent*) – possédant une ou plusieurs colonnes en commun. Pour la table « père », ces colonnes composent la clé primaire (ou candidate avec un index unique). Pour la table « fils », ces colonnes composent une clé étrangère.

Il est recommandé de créer un index par clé étrangère (Oracle ne le fait pas comme pour les clés primaires). La seule exception concerne les tables « pères » possédant des clés primaires (ou candidates) jamais modifiées ni supprimées dans le temps.

Cohérences

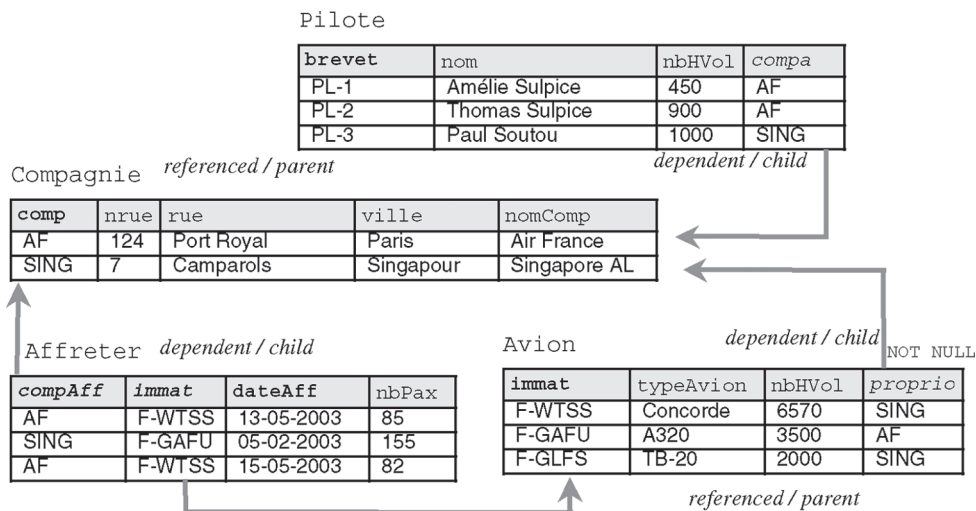
L'exemple suivant illustre quatre contraintes référentielles. Une table peut être « père » pour une contrainte et « fils » pour une autre (c'est le cas de la table Avion).



Deux types de problèmes sont automatiquement résolus par Oracle pour assurer l'intégrité référentielle :

- La cohérence du « fils » vers le « père » : on ne doit pas pouvoir insérer un enregistrement « fils » (ou modifier sa clé étrangère) rattaché à un enregistrement « père » inexistant. Il est cependant possible d'insérer un « fils » (ou de modifier sa clé étrangère) sans rattacher d'enregistrement « père » à la condition qu'il n'existe pas de contrainte NOT NULL au niveau de la clé étrangère.
 - La cohérence du « père » vers le « fils » : on ne doit pas pouvoir supprimer un enregistrement « père » (ou modifier sa clé primaire) si un enregistrement « fils » y est encore rattaché. Il est possible de supprimer les « fils » associés (DELETE CASCADE) ou d'affecter la valeur nulle aux clés étrangères des « fils » associés (DELETE SET NULL). Oracle ne permet pas de propager une valeur par défaut (*set to default*) comme la norme SQL2 le propose.
-

Figure 2-11 Tables et contraintes référentielles



Déclarons à présent ces contraintes sous SQL.

Contraintes côté « père »

La table « père » contient soit une contrainte de clé primaire soit une contrainte de clé candidate qui s'exprime par un index unique. Le tableau suivant illustre ces deux possibilités dans le cas de la table *Compagnie*. Notons que la table possédant une clé candidate aurait pu aussi contenir une clé primaire.

Tableau 2-14 Écritures des contraintes de la table « père »

Clé primaire	Clé candidate
<pre>CREATE TABLE Compagnie (comp VARCHAR2(4), nrue NUMBER(3), Rue VARCHAR2(20), ville VARCHAR2(15), nomComp VARCHAR2(15), CONSTRAINT pk_Compagnie PRIMARY KEY (comp));</pre>	<pre>CREATE TABLE Compagnie (comp VARCHAR2(4), nrue NUMBER(3), rue VARCHAR2(20), ville VARCHAR2(15), nomComp VARCHAR2(15), CONSTRAINT un_Compagnie UNIQUE(comp));</pre>

Contraintes côté « fils »

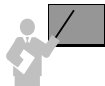
Indépendamment de l'écriture de la table « père », deux écritures sont possibles au niveau de la table « fils ». La première définit la contrainte en même temps que la colonne. Ainsi elle ne

convient qu'aux clés composées d'une seule colonne. La deuxième écriture détermine la contrainte après la définition de la colonne. Cette écriture est préférable car elle convient aussi aux clés composées de plusieurs colonnes de par sa lisibilité.

Tableau 2-15 Écritures des contraintes de la table « fils »

Colonne et contrainte	Contrainte et colonne
<pre>CREATE TABLE Pilote (brevet VARCHAR2(6) CONSTRAINT pk_Pilote PRIMARY KEY, nom VARCHAR2(15), nbHVol NUMBER(7,2), compa VARCHAR2(4) CONSTRAINT fk_Pil_compa_Comp REFERENCES Compagnie(comp));</pre>	<pre>CREATE TABLE Pilote (brevet VARCHAR2(6), nom VARCHAR2(15), nbHVol NUMBER(7,2), compa VARCHAR2(4), CONSTRAINT pk_Pilote PRIMARY KEY(brevet), CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY(compa) REFERENCES Compagnie(comp));</pre>

Clés composites et nulles



- Les clés étrangères ou primaires peuvent être définies sur trente-deux colonnes au maximum (*composite keys*).
- Les clés étrangères peuvent être nulles si aucune contrainte `NOT NULL` n'est déclarée.

Décrivons à présent les scripts SQL qui servent à notre exemple (la syntaxe de création des deux premières tables a été discutée plus haut) et étudions ensuite les mécanismes programmés par ces contraintes.

```
CREATE TABLE Compagnie ...

CREATE TABLE Pilote ...

CREATE TABLE Avion
(immat VARCHAR2(6), typeAvion VARCHAR2(15), nbHVol NUMBER(10,2),
proprio VARCHAR2(4),
CONSTRAINT pk_Avion PRIMARY KEY(immat),
CONSTRAINT nn_proprio CHECK (proprio IS NOT NULL),
CONSTRAINT fk_Avion_comp_Compag FOREIGN KEY(proprio)
REFERENCES Compagnie(comp));

CREATE TABLE Affreter
(compAff VARCHAR2(4), immat VARCHAR2(6), dateAff DATE,
nbPax NUMBER(3),
CONSTRAINT pk_Affreter PRIMARY KEY (compAff, immat, dateAff),
CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat)
REFERENCES Avion(immat),
CONSTRAINT fk_Aff_comp_Compag FOREIGN KEY(compAff)
REFERENCES Compagnie(comp));
```


Cohérence du fils vers le père



Si la clé étrangère est déclarée NOT NULL, l'insertion d'un enregistrement « fils » n'est possible que s'il est rattaché à un enregistrement « père » existant. Dans le cas inverse, l'insertion d'un enregistrement « fils » rattaché à aucun « père » est possible.

Le tableau suivant décrit des insertions correctes et une insertion incorrecte. Le message d'erreur est ici en anglais (en français : violation de contrainte d'intégrité - touche parent introuvable).

Tableau 2-16 Insertions correctes et incorrectes

Insertions correctes	Insertion incorrecte
<pre>-- fils avec père INSERT INTO Pilote VALUES ('PL-3', 'Paul Soutou', 1000, 'SING'); -- fils sans père INSERT INTO Pilote VALUES ('PL-4', 'Un Connu', 0, NULL); -- fils avec pères INSERT INTO Avion VALUES ('F-WTSS', 'Concorde', 6570, 'SING'); INSERT INTO Affreter VALUES ('AF', 'F-WTSS', '15-05-2003', 82)</pre>	<pre>-- avec père inconnu INSERT INTO Pilote VALUES ('PL-5', 'Pb de Compagnie', 0, '?'); ORA-02291: integrity constraint (SOUTOU.FK_PIL_COMPA_COMP) violated - parent key not found</pre>

Pour insérer un affrètement, il faut donc avoir ajouté au préalable au moins une compagnie et un avion.

Le chargement de la base de données est conditionné par la hiérarchie des contraintes référentielles. Ici, il faut insérer d'abord les compagnies, puis les pilotes (ou les avions), enfin les affrètements.



Il suffit de relire le script de création de vos tables pour en déduire l'ordre d'insertion des enregistrements.

Cohérence du père vers les fils

Trois alternatives sont possibles pour assurer la cohérence de la table « père » vers la table « fils » via une clé étrangère :

- Prévenir la modification ou la suppression d'une clé primaire (ou candidate) de la table « père ». Cette alternative est celle par défaut. Dans notre exemple, toutes les clés étrangères sont ainsi composées. La suppression d'un avion n'est donc pas possible si ce dernier est référencé dans un affrètement.

- Propager la suppression des enregistrements « fils » associés à l'enregistrement « père » supprimé. Ce mécanisme est réalisé par la directive `ON DELETE CASCADE`. Dans notre exemple, nous pourrions ainsi décider de supprimer tous les affrètements dès qu'on retire un avion.
- Propager l'affectation de la valeur nulle aux clés étrangères des enregistrements « fils » associés à l'enregistrement « père » supprimé. Ce mécanisme est réalisé par la directive `ON DELETE SET NULL`. Il ne faut pas de contrainte `NOT NULL` sur la clé étrangère. Dans notre exemple, nous pourrions ainsi décider de mettre `NULL` dans la colonne `compa` de la table `Pilote` pour chaque pilote d'une compagnie supprimée. Nous ne pourrions pas appliquer ce mécanisme à la table `Affreter` qui dispose de contraintes `NOT NULL` sur ses clés étrangères (car composant la clé primaire).

Tableau 2-17 Cohérence du « père » vers le « fils »

Alternative	Exemple de syntaxe
Prévenir la modification ou la suppression d'une clé primaire	<code>CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat) REFERENCES Avion(immat)</code>
Propager la suppression des enregistrements	<code>CONSTRAINT fk_Aff_na_Avion FOREIGN KEY(immat) REFERENCES Avion(immat) ON DELETE CASCADE</code>
Propager l'affectation de la valeur nulle aux clés étrangères	<code>CONSTRAINT fk_Pil_compa_Comp FOREIGN KEY(compa) REFERENCES Compagnie(comp) ON DELETE SET NULL</code>



L'extension de la modification d'une clé primaire vers les tables référencées n'est pas automatique (il faut la programmer si nécessaire par un déclencheur).

En résumé

Le tableau suivant résume les conditions requises pour modifier l'état de la base de données en respectant l'intégrité référentielle.

Tableau 2-18 Instructions SQL sur les clés

Instruction	Table « parent »	Table « fils »
<code>INSERT</code>	Correcte si la clé primaire (ou candidate) est unique.	Correcte si la clé étrangère est référencée dans la table « père » ou est nulle (partiellement ou en totalité).
<code>UPDATE</code>	Correcte si l'instruction ne laisse pas d'enregistrements dans la table « fils » ayant une clé étrangère non référencée.	Correcte si la nouvelle clé étrangère référence un enregistrement « père » existant.
<code>DELETE</code>	Correcte si aucun enregistrement de la table « fils » ne référence le ou les enregistrements détruits.	Correcte sans condition.
<code>DELETE CASCADE</code>	Correcte sans condition.	Correcte sans condition.
<code>DELETE SET NULL</code>	Correcte sans condition.	Correcte sans condition.

Exercices

Les objectifs des premiers exercices sont :

- d'insérer des données dans les tables du schéma *Parc Informatique* et du schéma des chantiers ;
- de créer une séquence et d'insérer des données en utilisant une séquence ;
- de modifier des données.

Exercice 2.1 Insertion de données

Écrivez puis exécutez le script SQL (que vous appellerez `insParc.sql`) afin d'insérer les données dans les tables suivantes :

Tableau 2-19 Données des tables

Table	Données					
Segment	INDIP	NOMSEGMENT	ETAGE			
	130.120.80	Brin RDC				
	130.120.81	Brin 1er étage				
	130.120.82	Brin 2ème étage				
Salle	NSALLE	NOMSALLE	NBPOSTE	INDIP		
	s01	Salle 1	3	130.120.80		
	s02	Salle 2	2	130.120.80		
	s03	Salle 3	2	130.120.80		
	s11	Salle 11	2	130.120.81		
	s12	Salle 12	1	130.120.81		
	s21	Salle 21	2	130.120.82		
	s22	Salle 22	0	130.120.83		
	s23	Salle 23	0	130.120.83		
Poste	NPOSTE	NOMPOSTE	INDIP	AD	TYPEPOSTE	NSALLE
	p1	Poste 1	130.120.80	01	TX	s01
	p2	Poste 2	130.120.80	02	UNIX	s01
	p3	Poste 3	130.120.80	03	TX	s01
	p4	Poste 4	130.120.80	04	PCWS	s02
	p5	Poste 5	130.120.80	05	PCWS	s02
	p6	Poste 6	130.120.80	06	UNIX	s03
	p7	Poste 7	130.120.80	07	TX	s03
	p8	Poste 8	130.120.81	01	UNIX	s11
	p9	Poste 9	130.120.81	02	TX	s11
	p10	Poste 10	130.120.81	03	UNIX	s12
	p11	Poste 11	130.120.82	01	PCNT	s21
	p12	Poste 12	130.120.82	02	PCWS	s21

Tableau 2-19 Données des tables (suite)

Table	Données					
Logiciel	NLOG	NOMLOG	DATEACH	VERSION	TYPELOG	PRIX
	log1	Oracle 6	13/05/95	6.2	UNIX	3000
	log2	Oracle 8	15/09/99	8i	UNIX	5600
	log3	SQL Server	12/04/98	7	PCNT	2700
	log4	Front Page	03/06/97	5	PCWS	500
	log5	WinDev	12/05/97	5	PCWS	750
	log6	SQL*Net		2.0	UNIX	500
	log7	I. I. S.	12/04/02	2	PCNT	810
	log8	DreamWeaver	21/09/03	2.0	BeOS	1400
Types	TYPELP	NOMTYPE				
	TX	Terminal X-Window				
	UNIX	Système Unix				
	PCNT	PC Windows NT				
	PCWS	PC Windows				
	NC	Network Computer				

Exercice 2.2 Gestion d'une séquence

Dans ce même script, créez la séquence `sequenceIns` commençant à la valeur 1, d'incrément 1, de valeur maximale 10 000 et sans cycle. Utilisez cette séquence pour estimer la colonne `numIns` de la table `Installer`. Insérez les enregistrements suivants :

Tableau 2-20 Données de la table `Installer`

Table	Données				
Installer	NPOSTE	NLOG	NUMINS	DATEINS	DELAI
	p2	log1	1	15/05/03	
	p2	log2	2	17/09/03	
	p4	log5	3		
	p6	log6	4	20/05/03	
	p6	log1	5	20/05/03	
	p8	log2	6	19/05/03	
	p8	log6	7	20/05/03	
	p11	log3	8	20/04/03	
	p12	log4	9	20/04/03	
	p11	log7	10	20/04/03	
	p7	log7	11	01/04/02	

Exercice 2.3 Modification de données

Écrivez le script `modification.sql`, qui permet de modifier (avec `UPDATE`) la colonne `etage` (pour l'instant nulle) de la table `Segment` afin d'affecter un numéro d'étage correct (0 pour le segment 130.120.80, 1 pour le segment 130.120.81, 2 pour le segment 130.120.82).

Diminuez de 10 % le prix des logiciels de type 'PCNT'.

Vérifiez :

```
SELECT * FROM Segment;
SELECT nLog, typeLog, prix FROM Logiciel;
```

Exercice 2.4 Insertion dans la base *Chantiers*

Écrivez puis exécutez le script SQL (que vous appellerez `insChantier.sql`) afin d'insérer les données suivantes :

- une dizaine d'employés (numéros E1 à E10) en considérant diverses qualifications (OS, Assistant, Ingénieur et Architecte) ;
- quatre chantiers et cinq véhicules ;
- deux ou trois visites de différents chantiers durant trois jours ;
- la composition (de un à trois employés transportés) de chaque visite.