ADMINISTRATION DU SYSTÈME ORACLE 10G

M. NEMICHE (IGE Semestre 5 Faculté Polydisciplinaire de Ouarzazate)

COMPOSITION DU MODULE

Matières

1 : Architecture et Installation Oracle

2 : Programmation sous oracle (PL/SQL)

3 : Les outils d'administration Oracle

3

Architecture Oracle

Architecture et Installation Oracle

4

- Objectifs
  - Installation d'Oracle
  - Comprendre l'architecture d'un serveur de BD Oracle
  - Démarrage et arrêt d'une instance et d'une base de données Oracle
  - Création d'une base de données opérationnelle
  - Gestion des fichiers d'une base de données Oracle
  - Gestion de la structure logique (tablespaces, segments, extents et blocs)

Bases de données relationnelles

5

- Collection de données opérationnelles enregistrées sur un support adressable et utilisées par les systèmes et les applications
- Les données doivent être structurées indépendamment d'une application particulière
- Elles doivent être cohérentes (contraintes), non redondantes (formes normales) et accessibles simultanément par plusieurs utilisateurs

SGBD : Système de Gestion de bases de données

6

- Un système de gestion de base de données (SGBD) est un ensemble de programmes qui permettent la gestion et l'accès à une base de données.
  - Un SGBD possède son propre système de fichier.
  - Un SGBD assure la reprise en cas de panne.
  - Un SGBD doit permettre la sauvegarde et la restauration d'une BD.
  - Un SGBD doit permettre une gestion des rôles et droits.
  - Une des fonctions importante des SGBD modernes est d'autoriser les utilisateurs d'effectuer des opérations simultanées (concurrentes) sur des données partagées de la BD. Si ces op ne sont pas sous contrôle, les accès interfèrent tôt ou tard les uns avec les autres et la BD devient incohérente. Pour éviter cela, le SGBD met en place un protocole de contrôle de simultanéité (ou de concurrence) qui empêche les accès à la BD d'interférer.

Transactions

7

- **Une transaction** est un ensemble de modifications de la base qui forme un tout indivisible. Il faut effectuer ces modifications entièrement ou pas du tout, sous peine de laisser la base dans un état incohérent.

Transactions

8

- Action **atomique** : entièrement ou pas du tout
- Préservant la **consistance** de la BD
- Comme si l'usager était **isolé** sur la BD : ses résultats intermédiaires (état temporairement incohérent) sont masqués aux autres transactions.
- A effet **durable** sur la BD, une fois terminées comme prévu
  - les effets d'une transaction globalement terminée ne peuvent pas être détruits ultérieurement par une quelconque défaillance.

Modèle ACID de transactions

Les tâches de l'administrateur de la Base de données

9

Les tâches de l'administrateur de la Base de données

10

- Dans la phase de « conception »
  - définition du schéma conceptuel de la base
  - règles de gestion, cohérence des informations
  - volumétrie
- Dans la phase de maintenance
  - Planification et création des BD
    - Gestion des structures physiques
    - Gestion des structures logiques
  - Gestion de la sécurité, des utilisateurs
  - Sauvegarde et restauration
  - Optimisation de la base de données
    - optimisation de requêtes
  - Administration du réseau

Les tâches de l'administrateur de la Base de données

11

- Grandes fonctions de DBA :
  - installer le SGBD et les applications clientes
  - Créer la base de données en faisant des choix au niveau physique
  - Gérer les utilisateurs
  - Assurer la cohérence et la sécurité des données
  - Echanger des données avec l'extérieur
  - Améliorer les performances
    - gestion des ressources mémoires
    - gestion des temps de réponses

Généralités

12

- Tendances actuelles
  - progiciels intégrés
    - minimise les besoins en administration
      - … sans pour autant les supprimer
  - amélioration des outils d'administration par les fournisseurs de SGBD
    - Notion d'Assistant
      - pour la création des bases, la sauvegarde/restauration, ...
  - A terme, vers des BD qui s'autoadministrent

Architecture Client Serveur

13

- L'architecture client/serveur désigne un mode de communication entre plusieurs ordinateurs à doubles niveaux d'hiérarchie.
- Le logiciel client peut envoyer des requêtes à un serveur via un protocole de communication à travers un support (réseau).
- Le serveur est initialement passif à l'écoute des requêtes clients sur un port déterminé. dès qu'une requête lui parvient, il décide de la traiter ou de la mettre en attente et envoie une réponse.
- Oracle est un SGBD doté d'une architecture Client/Serveur.

Histoire d'Oracle

14

- Software Development Laboratories (SDL) a été créé en 1977.
- En 1979, SDL change de nom en devenant Relational Software, Inc. (RSI) et introduit son produit Oracle V2 comme base de données relationnelle.
  - La version 2 ne supportait pas les transactions mais implémentait les fonctionnalités SQL basiques de requête et jointure. Il n'y a jamais eu de version 1, pour des raisons de marketing, la première version a été la version 2. Celle-ci fonctionnait uniquement sur les systèmes Digital VAX/VMS.

Histoire d'Oracle

15

- En 1983, RSI devient Oracle Corporation pour être plus représentative de son produit phare.
  - La version 3 d'Oracle, entièrement ré-écrite en langage de programmation C, est publiée.
  - Supporte les transactions grâce aux fonctionnalités de *commit* et *rollback*.
  - Unix est supportée dans cette version
- En 1984, la version 4 d'Oracle apparaît, supportant la cohérence en lecture (*read consistency*).
- Début 1985, Oracle commence à intégrer le modèle client-serveur, avec l'arrivée des réseaux au milieu des années 1980.

Histoire d'Oracle

16

- En 1988, Oracle met sur le marché son ERP - Oracle Financials basé sur la base de données relationnelle Oracle.
  - Oracle version 6 supporte le PL/SQL
  - le verrouillage de lignes (*row-level locking*)
  - les sauvegardes à chaud (*hot backups*, lorsque la base de données est ouverte).
- En 1992, la version 7 d'Oracle supporte les contraintes d'intégrité, les procédures stockées et les déclencheurs (*triggers*).
- En 1995, acquisition d'un puissant moteur multidimensionnel, commercialisé sous le nom d'Oracle Express.

Histoire d'Oracle

17

- En 1997, la version 8 introduit le développement orienté objet et les applications multimédia.
- En 1999, la version 8i est publiée dans le but d'affiner ses applications avec Internet. La base de données comporte nativement une machine virtuelle Java.
- En 2001, Oracle 9i
- En 2004, la version 10g est publiée.
- En 2005, vers la fin novembre, une version complètement gratuite est publiée, la « Oracle Database 10g Express Edition ».
- Septembre 2009, sortie de Oracle 11g Release 2

Généralités

18

- Oracle 10g est commercialisé selon trois gammes (Edition):
  - Edition Standard (Standard Edition)
  - Edition Entreprise (Entreprise Edition)
  - Edition Personnelle (Personal Edition)
- Oracle Express Edition

Généralités

19

- Oracle 10g Database est un SGBD qui fonctionne sur de nombreuse plates formes Unix(dont Linux) et également dans l'environnement Windows
  - Oracle propose donc une organisation de la mémoire et des ressources disque la plus indépendante possible de l'environnement système. C'est un SGBD qui offre une architecture très ouverte.

Généralités

20

L'architecture Oracle comporte plusieurs composants principaux :

- Serveur Oracle: comporte plusieurs fichiers, processus et structures mémoire. Le serveur Oracle est constitué d'une instance oracle et d'une base oracle
  - Instance Oracle: L'instance Oracle comprend une région de la mémoire appelée La SGA (System Global Area), ainsi que les processus d'arrière plan utilisé pour gérer la base de données

Généralités

21

- La zone mémoire du programme (PGA)
  - Zone mémoire utilisée par un seul processus serveur à la différence de la SGA qui est partagée par tous les processus serveurs
  - PGA contient :
    - une zone de tri
    - des informations sur la session
    - l'état du curseur
    - …

Généralités

22

- Base de données Oracle:
  - Structure physique:
    - Fichiers de données, Fichiers redo log, Fichiers de contrôle.
    - Autres fichiers importants: (fichier de paramètres, fichier de mots de passe)
  - Structure logique
    - Tablespace, segment, extent, bloc
- Les Processus serveurs: gèrent les requêtes des utilisateurs provenant des connexions à la base de données; ils sont chargés de:
  - la communication entre la SGA et le processus utilisateur.
  - analyser, d'exécuter les requêtes SQL des utilisateurs, de lire les fichiers de données, de placer les blocs de données correspondants dans la SGA et de Renvoyer les résultats des commandes SQL au processus utilisateur.

Généralités

23

- Le serveur oracle supporte
  - SQL (LDD , LMD, LCD)
  - PL/SQL
  - Autres langages de programmation (Pro*C …)

Généralités

24

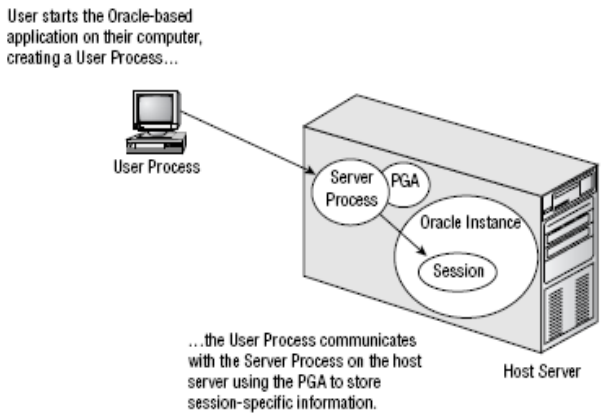
- **Connexion a un serveur Oracle**
  - Une connexion est un chemin de communication entre un processus utilisateur et un processus serveur. Il existe trois types de connexions grâce auxquelles un utilisateur peut accéder à un Serveur Oracle :
    - Connexion locale : Selon cette méthode, un utilisateur est directement connecté sur la machine faisant office de Serveur Oracle.
    - Connexion Deux Tiers : Ce type de connexion est couramment nommé "Connexion Client Serveur", un utilisateur se connecte à partir d'une machine directement connectée à un Serveur Oracle.
    - Connexion Multi Tiers : Dans une architecture multi tiers, la machine de l'utilisateur se connecte à un Serveur applicatif (Par exemple un Serveur Web) qui lui même va se connecter au serveur Oracle pour récupérer les données issues de la base de données.

Généralités

25

- Session: une session est une connexion spécifique d'un utilisateur à un serveur oracle.
- La session démarre lorsque l'utilisateur est authentifié par le serveur oracle et se termine lorsque l'utilisateur se déconnecte ou en cas de déconnexion anormale.

The relationship between User and Server processes



The Server Process communicates with the Oracle instance on behalf of the user. The Oracle instance is examined in the next section.

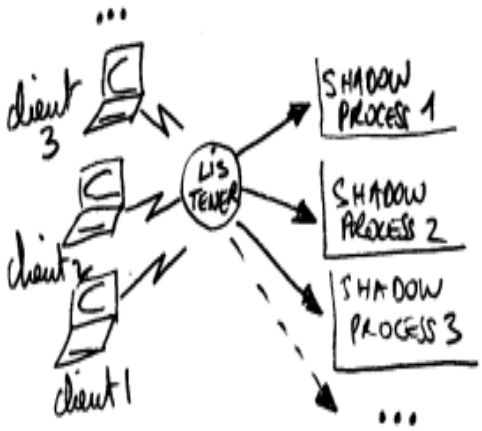
26

Généralités

27

- Oracle supporte deux mode de fonctionnement:
  1. Serveur dédié : chaque fois qu'un utilisateur se connecte, il est pris en charge par un processus serveur.
     - Si 100 utilisateurs se connectent, 100 processus serveurs sont crées de même
     - Avantage:
       - Une commande SQL est tout de suite et directement prise en compte par un processus serveur
     - Inconvénient:
       - Chaque processus serveur occupe une zone mémoire et utilise la CPU
     - Meilleure configuration (recommandée et utilisée par la bcp de DBA), si les ressources matérielles le permettent.

28

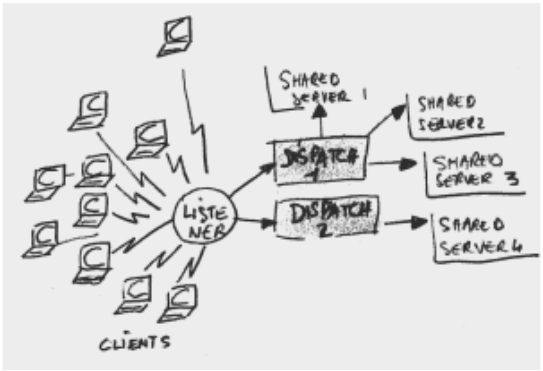


Généralités

29

2. Serveur Partagé : c'est un groupe de processus serveurs qui s'occupent d'un grand nombre de processus utilisateurs.
   - Les processus utilisateurs sont alloués à un processus DISPACHER, celui-ci met les requêtes utilisateurs dans une fille d'attente, et le processus serveur exécute toutes les requêtes, une par une
   - Avantage:
     - Réduire la charge de la CPU et utilise moins de mémoire
   - Inconvénient:
     - Lors de forte utilisations de la BDD, il risque d'y avoir des temps d'attente (↓ performance)

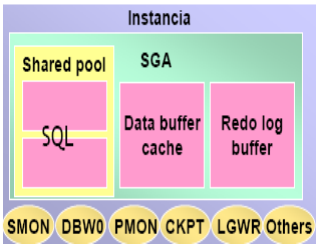
30



Architecture - Instance

31

- Serveur Oracle = instance Oracle + base de données Oracle
  - Instance Oracle :
    - c'est un moyen pour accéder à une base de données Oracle (ouvre une unique base de données)
      1. Structure Mémoire (SGA)
      2. Processus en arrière plan



L'utilisation de la mémoire par Oracle 10g

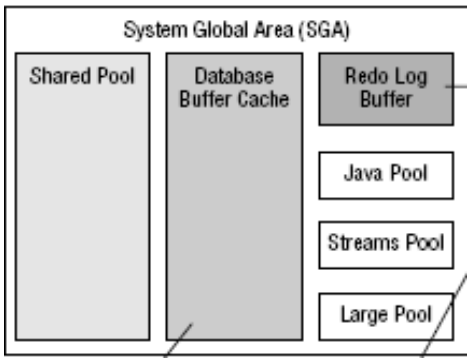
32

- Dans tout système informatique, l'utilisation de mémoire est synonyme de performance.
- la mémoire partagée SGA (*System Global Area)*:
- la mémoire allouée pour chaque programme PGA (*Program Global Area) ;*
- Les données auxquelles on accède et qui sont manipulées en mémoire le sont beaucoup plus rapidement que sur disque.
- Il est important de bien comprendre ces éléments, car ils interviennent dans les opérations d'amélioration des performances.

Architecture - Instance : SGA

33

- La SGA (*System Global Area) représente la zone mémoire déterminante d'une instance, tant* par sa taille que par son rôle.
  - C'est elle qui assure le partage des données entre les utilisateurs.
    - Oracle utilise la mémoire SGA comme buffer intermédiaire (plus rapide que le disque) pour l'échange de données entres processus
  - Elle est divisée trois composants obligatoires :
    - shared pool
    - Database buffer cache (Le cache de données)
    - redo log buffer (Le cache de reprise)
  - Et de trois composants optionnels
    - Java pool
    - Large pool
    - Streams pool



34

Architecture - Instance : SGA

35

Database Buffer cache

- Est utilisé pour stocker des blocs de données en mémoire afin d'accélérer l'interrogation et/ou la modification
- Aucune modification est faite directement sur les données du disque
  - Oracle lit les données suite à la demande d'un *processus utilisateur et ensuite valide les* modifications sur le disque
- Il utilise un algorithme nommé LRU mois récemment utilisés (Least-Recently Used) pour déterminer les données à libérer du cache

Architecture - Instance : SGA

36

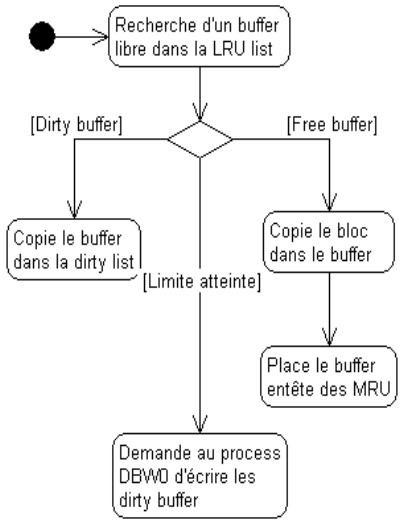
Organisation du buffer cache

- Le buffer cache est organisé en 2 listes :
  - La dirty list
  - La liste LRU (Least Recently Used)
- La dirty list contient les buffers qui ont été modifiés mais ne sont pas encore écrits sur le disque
- La liste LRU contient les buffers libres (qui n'ont pas été modifiés), les « pinned » buffers (qui sont en cours de modification, les dirty buffers qui n'ont pas encore été déplacés dans la dirty list

Architecture - Instance : SGA

Database Buffer cache (recherche d'un buffer libre)

37



Architecture - Instance : SGA

Database Buffer cache

38

- Vues système utilisées
  - V$SGA ;
  - V$PARAMETER ;

Architecture - Instance : SGA

39

Buffer Redo Log

- C'est un buffer circulaire qui stocke les modifications réalisées sur la base de données avec les opérations: insert, delete, update, create, alter y drop.
- Permet à oracle de reconstruire les modifications des données en cas de panne
- L'information Redo reste dans le buffer Redo log jusqu'à ce qu'oracle la stocke sur le disque
- Sa taille est définie par LOG_BUFFER

Architecture - Instance : SGA

40

Shared Pool

- Permet de stocker plusieurs éléments cruciaux pour la gestion des données :
  - Library cache : permet d'analyse l'ordre d'exécution d'une requête SQL et de définir un plan d'exécution.
  - Dictionary cache : stocke des les données en provenance du dictionnaire afin d'accélérer l'accès au dictionnaire (nom d'utilisateurs, privilèges, etc.).
  - SQL area : stocke les requêtes SQL les plus récemment utilisées par les utilisateurs de base de données.
    - Si la même requête est ré-exécutée, le serveur n'analyse pas son ordre. Cela permet d'améliorer la performance des applications

Architecture - Instance : SGA

41

- Le fonctionnement d'une base Oracle est assuré par un ensemble de processus imbriqués qui réalisent de nombreuses actions. Pour plus de simplicité, nous avons regroupé les processus en deux familles :
  - les indispensables,
  - les optionnels.

Architecture - Instance : Processus en arrière plan

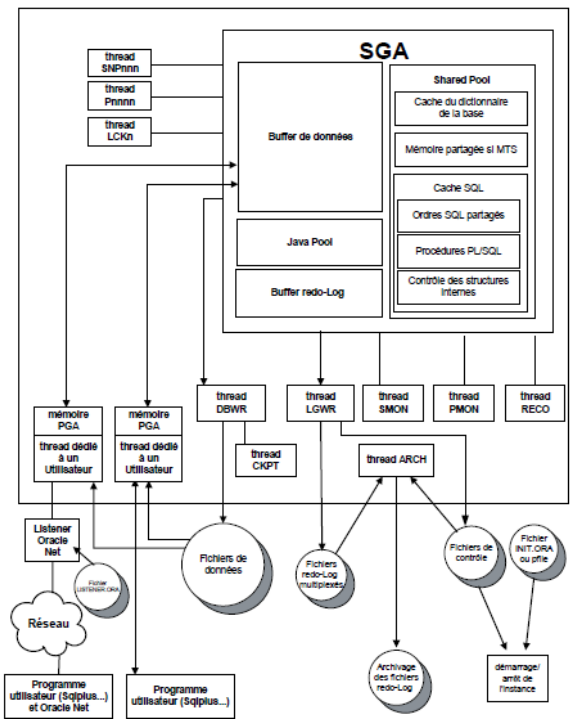
42

- Les processus indispensables sont présents dès qu'une base Oracle fonctionne. Ils sont requis pour en assurer le fonctionnement minimal. Si l'un d'eux s'arrête, la base de données n'est plus opérationnelle.
- D'autres processus peuvent être lancés pour assurer des fonctions complémentaires, comme l'archivage des redo log. Si l'un de ces processus optionnels n'est pas démarré, cela ne met pas en cause le fonctionnement global de la base de données. Seule la tâche assurée par ce processus optionnel ne sera pas réalisée.

Les processus indispensables

43

- DBWR (*Database Writer*) ;
- LGWR (*Log Writer*) ;
- CKPT (*Checkpoint*) ;
- PMON (*Process Monitor*) ;
- SMON (*System Monitor*).



44

Les processus indispensables

45

- DBWR (Data Base WRiter)
  - Ecrit les blocs modifiés dans le cache de données sur les disques.
  - DBWn se déclenchera lors des événements suivants :
    - Lorsque le nombre de bloc *dirty* atteint une certaine limite
    - Lorsqu'un processus sera à la recherche de blocs libres dans le *Database Buffer Cache*, et qu'il ne sera pas en mesure d'en trouver.
    - Lors de timeouts (environ toutes les 3 secondes par défaut)
    - Lors d'un *checkpoint*.

Les processus indispensables

46

- Le comportement de DBWR est contrôlé par le paramètre d'initialisation DB_WRITERS, qui permet de démarrer plusieurs processus DBWR, afin d'augmenter le taux d'écriture sur disque dans les systèmes très fortement sollicités. Nous vous conseillons de démarrer une base Oracle 10g *avec un seul processus DBWR et d'en augmenter progressivement le* nombre dans les systèmes très sollicités en entrées/sorties disque.

Les processus indispensables

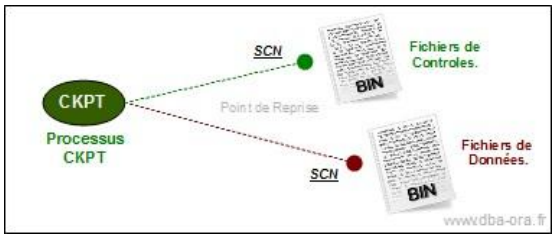
47

- LOGWR (LoG WRiter)
  - Ecrit dans les fichiers Redo Log le contenu du cache Redo Log
  - Le processus LGWR transcrit les informations contenues dans le *REDO LOG Buffer* vers les fichiers *REDOLOG FILE* quand :
    - une transaction s'est terminée avec un COMMIT
    - le REDO LOG Buffer est au 1/3 plein
    - Avant que DBWn n'écrive le contenu du Database Buffer Cache dans les fichiers du disque dur

Les processus indispensables

48

- CKPT (CHECKPOINT)
  - **Checkpoint** inscrit les informations de point de reprise dans les fichiers de Controles et dans l'entête de chaque fichier de données. C'est ce point de reprise (**SCN**) qui permet de rendre cohérent les fichiers de controles et les fichiers de données, indispensable pour un processus de récupération.



Les processus indispensables

49

- CKPT (CHECKPOINT)
  - Le processus Checkpoint n'écrit pas les blocs sur le disque, c'est le role du processus Database Writer DBWn.
    - Provoque l'activation du DBWR pour écrire les blocs modifiés depuis le dernier point de contrôle,
  - Les numéros SCN enregistrés dans les fichiers garantissent que toutes les modifications apportées aux blocs de base de données avant un numéro SCN ont été écrites sur le disque.
    - En cas d'arrêt anormal de l'instance, ce SCN marque le début des données à utiliser pour la récupération de l'instance.

Les processus indispensables

50

- CKPT (CHECKPOINT)
  - Le processus Log Writer (LGWR) n'écrit pas dans le fichier de journalisation (REDO LOG) tant que le processus CKPT n'a pas synchronisé, car tant que cette synchronisation n'est pas faite, le fichier de journalisation contient des données nécessaires à une éventuelle récupération après défaillance de l'instance.
  - A savoir qu'une synchronisation se déclenche aussi à chaque basculement de REDO LOG, lors de l'arrêt de la base de données, lors de la mise hors ligne d'un Tablespace.

Les processus indispensables

51

- SMON (System MONitor)
  - **System Monitor** surveille la base de données lors de son démarrage c'est à dire faire la récupération de l'instance après un arrêt anormal. Lors du démarrage de l'instance Oracle, il vérifie si le dernier arrêt a été correctement effectué.
  - **SMON** est aussi chargé de:
    - libérer les segments temporaires
    - compacter l'espace contigu dans les Tablespaces
    - surveiller la base de données.

Les processus indispensables

52

- Le processus SMON en action lors d'un arrêt brutal de la base de données
  - **SMON** lit les informations contenu dans les segments **UNDO** (données en attente de validation) puis les annule. (**ROLL BACK**)
  - **SMON** récupère dans les fichiers **REDO LOG** les enregistrements validés mais pas encore écrit dans les fichiers de données et les insère. (**ROLL FORWARD**).
  - **SMON** libère toute les ressources de la base de données (vérrous, segments temporaires).

Les processus indispensables

53

- Le **processus System Monitor SMON** en action lors d'un fonctionnement normal de la base de données
  - **SMON** surveille l'activité de la base de données.
  - **SMON** recycle les segments temporaires et assure les espaces de tris.
  - **SMON** libère toute les ressources de la base de données (vérrous, segments temporaires).
  - **SMON** peut être appelé par d'autre Processus pour libérer de l'espace.

Les processus indispensables

54

- PMON (Process Monitor)
  - Ce processus gère les ressources utilisateurs
  - Si un incident survient (arrêt brutal du poste client)
    - PMON annule les transactions en cours
    - Il supprime les verrous posés sur les tables et les lignes
    - Et libère les ressources utilisées

Les processus optionnels

55

- Les processus détachés optionnels sont les suivants :
- Listener ou listener Net ;
- ARCH ou Archiver ;
- RECO ou Recover ;
- Dnnnn et Snnnn Dispatcher et Server ;
- ....

Les processus server

56

- Le rôle des processus server est fondamental. Dès que l'application demande l'accès à l'instance, ce sont eux qui vérifient que les données sont présentes en mémoire.
  - Dans l'affirmative, le processus server traite vos ordres SQL.
  - Dans la négative, les processus server lisent les fichiers de la base de données pour les « monter » en mémoire SGA, avant de traiter l'ordre SQL.

Les processus server

57

- Ce n'est pas un processus permanent de l'instance
  - le processus server est créé lors de chaque connexion
- Il assure la lecture des données contenues dans vos fichiers.
- Il assure le lien entre le processus client, les processus d'arrière plan, la SGA et les fichiers qui composent la base.

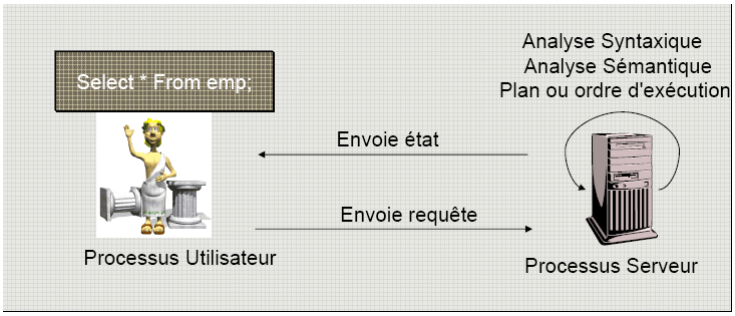
Traitement d'une requête d'interrogation

58

- Le processus client envoie la requête au processus serveur
- Le processus serveur effectue l'analyse syntaxique et sémantique de la requête
  - utilise la shared pool de la SGA pour compiler l'ordre
  - retourne l'état (analyse correcte ou incorrecte) au processus client.
- Exécution de la requête
- Récupération des résultats
  - le processus serveur envoie les lignes extraites par la requête (à partir du cache de données si les données y ont déjà été chargées)

Traitement d'une requête d'interrogation

59



Traitement d'une requête de MAJ

60

- Similaire à une requête d'interrogation pour la phase d'analyse
- Phase d'exécution différente :
  1. le processus serveur lit des blocs de données et de rollback à partir
     - des fichiers de données
     - du buffer cache de données
  2. Si des blocs ne sont pas déjà dans le cache, copie des blocs du disque dans le cache
  3. Mise à jour de verrou sur les données

Traitement d'une requête de MAJ

61

4. Enregistrement des modifications
   - à apporter au rollback (image avant)
   - et aux données (nouvelles valeurs) dans le cache de reprise (redo log)
5. Mêmes opérations dans le cache de données
   - ces blocs sont marqués comme modifiés (différents des blocs stockés sur disque)

Architecture - BDD

62

- Base de Données Oracle
  - Structure Physique de stockage
    - Fichiers de données
    - fichiers Redo Log
    - Fichiers de Contrôles
    - Autres fichiers
  - Structure Logique de stockage
    - Tablespace
    - Segment
    - Extent
    - Bloc de données

➔ Nécessaire de comprendre ces 2 niveaux de représentation

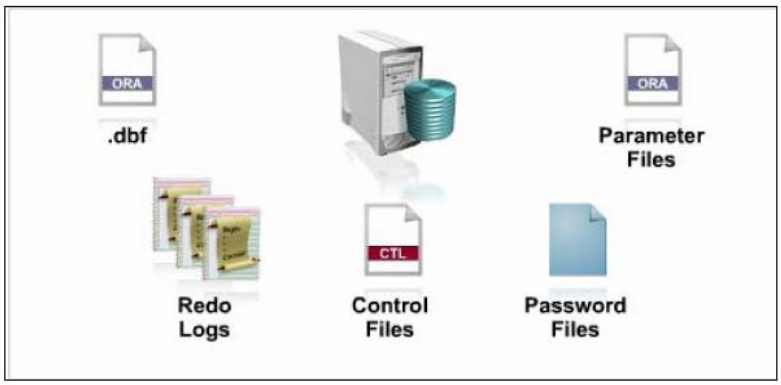
BD Oracle : Rappel

63

- se compose de fichiers du SE
  - fichiers de données (>=2)
    - stocke le dictionnaire des données, les objets utilisateurs …
  - fichiers de reprise (redo log) (>=2)
    - Stocke toutes les modifications apportées à la base de données (pour la reconstruire en cas de panne)
  - fichiers de contrôle (>=1)
    - contient les informations nécessaires à la mise à jour et à la vérification de l'intégrité des données.
  - Autres fichiers :
    - fichier de paramètres, fichier mot de passe, fichiers de reprise archivés …

Architecture – structure Physique de la BDD

64



Architecture – structure Physique de la BDD

65

- Fichiers de données

Architecture – structure Physique de la BDD

66

select * from dba_data_files;

Présentation du dictionnaire de données

67

- Le dictionnaire de données est un ensemble de tables et de vues où est stockée l'information sur la structure logique et physique de la BDD:
  - Les utilisateurs des bases de données
  - Noms et caractéristiques des objets stockés dans la base de donnée
  - Contraintes d'intégrités
  - Ressources physiques allouées à la base
  - …

Présentation du dictionnaire de données

68

- Le dictionnaire de données est créé à la création de la BDD, mis à jour au fur et à mesure de la création d'objets.
- Les utilisateurs des bases de données, les développeurs d'applications, les administrateurs de bases de données et le serveur Oracle utilisent le dictionnaire de données comme une source centrale d'information associée à une base de données.

Présentation du dictionnaire de données

69

- Le dictionnaire de données possède deux composants :
  1. les tables de base
  2. les vues du dictionnaire de données

Présentation du dictionnaire de données

70

1. Les tables de base sont les tables réelles d'Oracle qui stockent les informations sur une base de données.
   - Ces tables sont créées avec le script sql.bsq. Ce script est stocké dans le répertoire ORACLE_HOME\rdbms\admin. Les informations stockées dans les tables de base sont lues et écrites par le serveur Oracle. Ces informations sont rarement accédées directement par les utilisateurs car ces informations sont normalisées et stockées sous une forme encodée.
   - Les utilisateurs ou administrateurs de bases de données ne doivent pas utiliser de commandes LMD, telles que INSERT, UPDATE et DELETE, pour mettre à jour les tables de base directement, à l'exception de la table de trace d'audit lorsque la fonctionnalité d'audit est utilisée.

Présentation du dictionnaire de données

71

2. Les vues du dictionnaire de données sont des vues sur les tables de base.
   - Elles sont crées par le script catalog.sql.
   - Les vues du dictionnaire de données simplifient et résument les informations contenues dans les tables de base.
   - Les vues du dictionnaire stockent également ces informations sous une forme que les utilisateurs de la base de données peuvent lire facilement.
   - Ces vues permettent au DBA de gérer et d'administrer la base de données.

Présentation du dictionnaire de données

72

- Les utilisateurs n'ont pas accès à ce dictionnaire éventuellement en le lecture à traves trois catégories des vues:
  1. USER_<vues> : Vues sur les objets créé par un utilisateur
     - Par exemple, la vue USER_TABLES contient les informations sur les tables appartenant à un utilisateur.

Présentation du dictionnaire de données

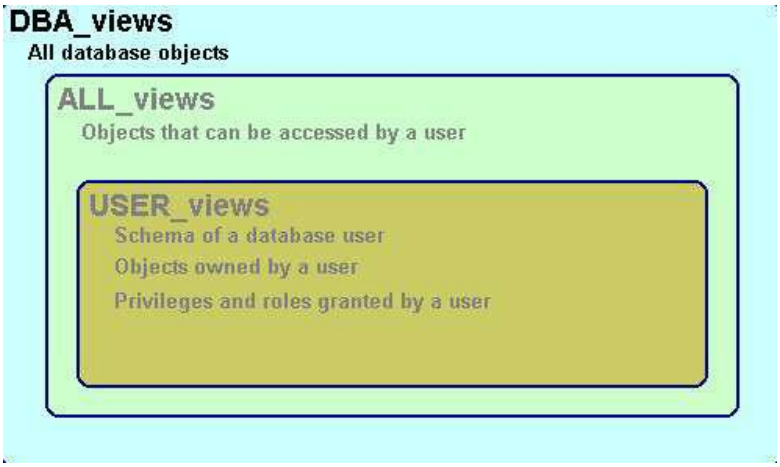
73

2. ALL_<vues>: Vues sur les objets auxquels un utilisateur a accès (pas nécessairement qu'il a créés(à travers l'obtention publique ou explicite de rôles et de privilèges)

3. DBA_<vues>: Vues sur tous les objets de la base de données de tous les utilisateurs accessible par les administrateur(DBA) ou les utilisateurs qui possèdent le privilège système SELECT ANY TABLE.

Présentation du dictionnaire de données

74



Présentation du dictionnaire de données

75

Vues	Description
dictionary dict_columns	Vues générales
dba_tables dba_objects dba_lobs dba_tab_columns dba_constraints	Informations sur les objets, tels que les tables, les contraintes, les gros objets et les colonnes
dba_users dba_sys_privs dba_roles	Informations sur les privilèges et les rôles des utilisateurs
dba_extents dba_free_space dba_segments	Allocation d'espace pour les objets de la base de données
dba_rollback_segs dba_data_files dba_tablespaces	Structures générales de la base de données
dba_audit_trail dba_audit_objects dba_audit_obj_opts	Informations d'audit

Architecture – structure Physique de la BDD

76

- fichiers Redo Log

Les fichiers Redo Logs

Ils sont utilisés pour stocker les informations Redo sur le disque afin de garantir la reconstruction des données en cas de panne

Une BDD Oracle requiert au moins 2 fichiers redo log

fichiers Redo Log

77

- L'idée de base enregistrer toutes les modifications apportées aux données pour minimiser les problèmes liés aux pannes
- Le serveur Oracle met à jour les fichiers de reprise
  - buffer de reprise
  - processus LGWR
- Ne sont utilisés qu'en cas d'échec d'une instance pour restaurer des données validées non écrites dans les fichiers de données

fichiers Redo Log

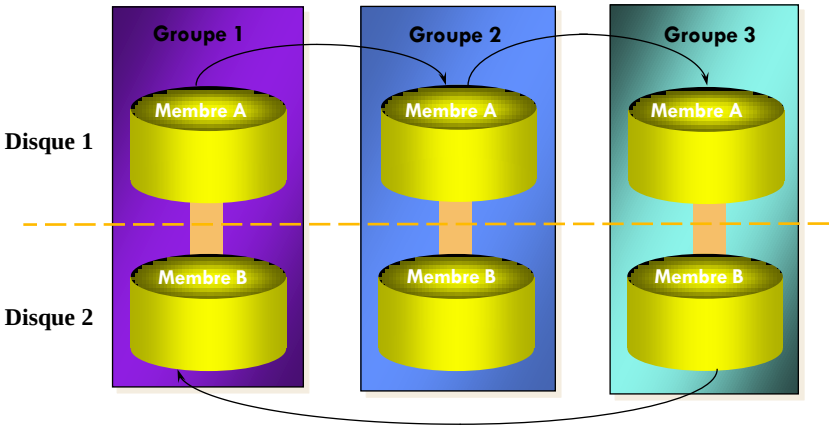
78

- Groupes de fichiers de reprise :
  - un membre est un élément d'un groupe, i.e. un fichier de reprise
  - LGWR écrit simultanément sur chaque membre du groupe
  - tous Les membres d'un groupe redo log en ligne possèdent exactement les mêmes informations
  - Les membres d'un groupe sur des disques différents (au moins 2 groupes)
  - tous les membres d'un groupe possèdent un numéro de séquence log sert d'identifiant
  - le numéro courant est stocké dans le fichier de contrôle

```
select * from v$logfile;
select * from v$log;
```

Les Fichiers Redo Log Multiplexés

79



fichiers Redo Log

80

- Ajouter un groupe de fichiers de reprise online:

ALTER DATABASE [*database*]
ADD LOGFILE [GROUP *valeur*] *'filename' [SIZE n[K|M]] [REUSE]*
[,[GROUP *valeur*] *'filename' [SIZE n[K|M]] [REUSE]]...;*

Exemple

```
SQL> ALTER DATABASE
        ADD LOGFILE GROUP 3
        ('c:\orant\database\logorcl3.ora') SIZE 1000K;
```

fichiers Redo Log

81

- **Ajout des membres redo log online**
  - Des membres redo log peuvent être ajoutés grâce à la commande SQL suivante :

    ALTER DATABASE *[database]*
    ADD LOGFILE MEMBER '*filename' [REUSE]*
    TO GROUP *n;*

    Exemple

```
SQL> ALTER DATABASE
       ADD LOGFILE MEMBER 'e:\orant\database\log7borcl.ora'
         TO GROUP 7;
```

fichiers Redo Log

82

- **Suppression de groupes de fichiers redo log online**
  - Pour améliorer les performances de la base de données, il peut s'avérer nécessaire d'augmenter ou de diminuer la taille des groupes de fichiers redo log online. Pour changer la taille d'un groupe de fichiers redo log online, il faut créer un nouveau groupe de fichiers redo log online et ensuite supprimer le vieux groupe de fichiers redo log online.

fichiers Redo Log

83

- **Suppression de groupes de fichiers redo log online**

  ALTER DATABASE [*database*]
  DROP LOGFILE GROUP *n;*

fichiers Redo Log

84

- Un certain nombre de restrictions sont à prendre en compte lors de la suppression de groupes redo log online :
  - L'instance doit avoir au moins deux groupes de fichiers redo log online. Un groupe de fichiers redo log online ne pourra pas être supprimé si il n'existe que deux groupes. Pour supprimer un groupe, il doit en rester au moins trois.
  - Un groupe redo log online actif ne peut pas être supprimé.
- Quand un groupe redo log est supprimé, les fichiers du système d'exploitation ne sont pas supprimés automatiquement. Il est donc nécessaire de supprimer manuellement les fichiers redo log online afin de garder un espace disque propre.

fichiers Redo Log

85

- **Emplacement des fichiers redo log online**
  - La disponibilité des fichiers redo log online détermine la performance d'une base de données Oracle.
    - Pour améliorer la performance de la base de données, les membres des groupes de fichiers redo log online, les fichiers log archivés et les fichiers de données doivent être stockés sur des disques séparés.
    - Le stockage des fichiers redo log et des fichiers de données sur différents disques permet de réduire de façon substantielle le risque de perdre à la fois les fichiers de données et les fichiers redo log online lors d'une défaillance du support.

fichiers Redo Log

86

- **Archivage de fichiers de redo log**
  - Une des décisions importantes qu'un DBA doit prendre est configurer la base de données dans le mode :
    1. ARCHIVELOG
    2. NOARCHIVELOG

fichiers Redo Log

87

- **Archivage de fichiers de redo log**
  - **NOARCHIVELOG :**
    - En mode, NOARCHIVELOG, les fichiers de redo log online sont réécrits à chaque fois qu'un fichier de redo log online est rempli et qu'un log switch est lancé.

fichiers Redo Log

88

- **Archivage de fichiers de redo log**
  - **ARCHIVELOG :**
    - Si la base de données est configurée dans le mode ARCHIVELOG, alors les groupes de redo remplis doivent être archivé. Comme toutes les modifications faites sur la base de données sont enregistrées dans les fichiers de redo log online, le DBA peut utiliser la sauvegarde présente sur le disque dur et les fichiers de redo log archivés pour restaurer la base de données sans perdre aucune donnée comité.
  - On peut archiver les fichiers de redo log :
    - Manuellement.
    - Automatiquement : Méthode recommandée.

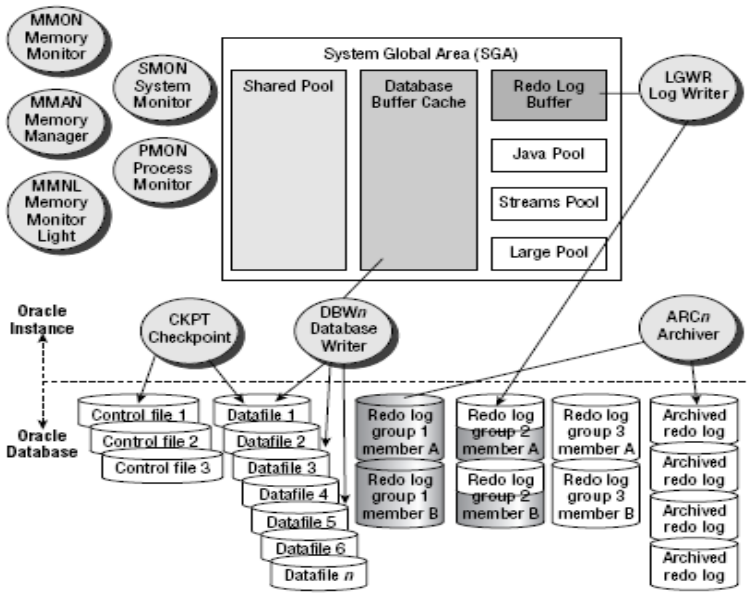
fichiers Redo Log

89

- **Archivage de fichiers de redo log**
  - **ARCHIVELOG :**
    - Le paramètre d'initialisation LOG_ARCHIVE_START, lorsqu'il est à
      - TRUE indique que l'archivage se fait automatiquement.
      - A FALSE, indique que le DBA le fait manuellement.
    - En mode automatique, l'archivage se fait grâce au processus ARCn, et manuellement avec des requêtes SQL.

90

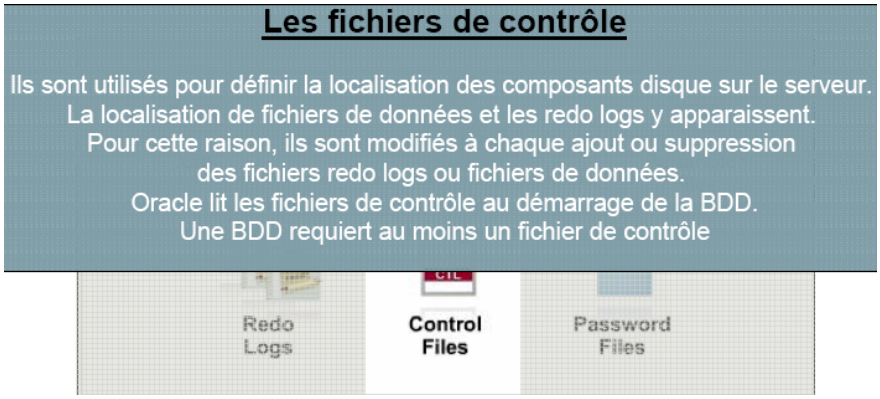
The Oracle 10*g* architecture



Architecture – structure Physique de la BDD

91

- Fichiers de Contrôles



Fichiers de Contrôles

92

- Fichier binaire
- Sert pour la base de données
  - lors du démarrage normal
    - A chaque fois qu'une instance monte une base de données, elle lit le fichier de contrôle pour localiser emplacement des fichiers de données et des fichiers de reprise
  - lors du démarrage après panne
    - Contient les informations nécessaires à la remise en état de la base de données
  - pour le bon fonctionnement
    - doit être disponible quand une base de données est montée ou ouverte

Fichiers de Contrôles

93

- Tous les fichiers de données et les fichiers log de la base de données sont identifiés dans le fichier de contrôle.
- Le nom de la base de données est répertorié dans le fichier de contrôle.
- Le fichier de contrôle est requis pour monter, ouvrir, et accéder à la base de données.
- Le numéro de séquence de log actuel. Cette information permet de synchroniser tous les fichiers appartenant à la base de données.
- La configuration recommandée est un minimum de deux fichiers de contrôle sur des disques différents (multiplixage).

Fichiers de Contrôles

94

- **Multiplexage des fichiers de contrôle**
  - Dans le but de prévenir une erreur dans un fichier de contrôle, il est fortement recommandé de multiplexer les fichiers de contrôles et de les stocker séparément sur des disques différents.
    - Si un fichier de contrôle est perdu, une copie du fichier de contrôle peut être utilisé pour redémarrer l'instance. On peut multiplexer jusqu'à 8 fichiers de contrôles.

Fichiers de Contrôles

95

- **Multiplexage des fichiers de contrôle**
  - Modifier le SPFILE (Fichiers de paramètres) :

    ALTER SYSTEM SET control_files =

    '$HOME/ORADATA/u01/ctrl01.ctl',

    '$HOME/ORADATA/u02/ctrl02.ctl' SCOPE = SPFILE ;
  - Eteindre la base de données.
  - Créer les fichiers de contrôles supplémentaires :

    cp $HOME/ORADATA/u01/ctrl01.ctl $HOME/ORADATA/u02/ctrl02.ctl
  - Redémarrer la base de données

Multiplexage du fichier de contrôle

96

- **Avec le init.ora** (Fichiers de paramètres) **:**
  - Eteindre la base de données.
  - Copier le fichier de contrôle existant sur un autre disque et changer sont nom :

    cp control01.ctl …/DISK3/control02.ctl
  - Ajouter le nouveau fichier de contrôle à init.ora :

    CONTROL_FILES = (/DISK1/control01.ctl,
    /DISK3/control02.ctl)
  - Redémarrer la base de données.

Fichiers de Contrôles

97

- Requêtes sur le fichier de contrôle
  - Select * from V$CONTROLFILE;

Architecture - BDD

98

- Autres fichiers
  - Fichiers de paramètres d'initialisation

Structure logique de stockage

99

- Objectifs
  - Faire le lien entre structure physique et logique
    - via les fichiers de données
  - Comprendre les moyens logiques de structuration
    - Segment
  - Etude de deux segments particuliers
    - Segment rollback
    - Segment temporaires

Tablespace : Définitions

100

- L'utilisateur, qu'il soit connecté par une application, ou directement à la BDD par SQL*Plus, ne voit que la structure logique de la BDD.
  - Quand il envoie une requête il voit que les tables et leurs description
  - A aucun moment ne fait appel dans sa commande aux fichiers physiques
  - Le processus serveur qui réceptionne la commande va devoir savoir dans qu'il fichier sont stockées
  - Le relation entre les tables et les données stockées dans le fichier physique se fait par un objet logique : Le tablespace

Tablespace : Définitions

101

- Les données d'une base Oracle sont mémorisées dans une ou plusieurs unités logiques appelées tablespaces et physiquement dans des fichiers associés à ces tablespaces.
- Un tablespace est la plus grande unité logique de stockage allouable dans oracle.
- Un tablespace est composé d'au moins un fichier de données ayant une existence physique(stocké sur disque dur).

Tablespace : Définitions

102

- Tout objet du schéma de base de données est crée dans un seul tablespace, mais peut s'étaler sur plusieurs fichiers de données.
- Les tablespaces peuvent être activés (online) ou désactivés (offline) individuellement pour d'éventuelles opérations de maintenance:
  - Les objets d'un tablespace offline ne sont plus accessibles.

Utilisation des Tablespaces

103

- Contrôle de l'allocation d'espace et affectation de quotas aux utilisateurs
- Contrôle de disponibilité des données par la mise online ou offline des tablespaces
- Distribution du stockage des données sur plusieurs dispositifs
  - pour améliorer les E/S
  - pour réduire la contention sur un seul disque
- Exécution de sauvegarde partielle
- Conservations de volumes importants de données statiques sur des dispositifs en lecture seule.

Tablespace

104

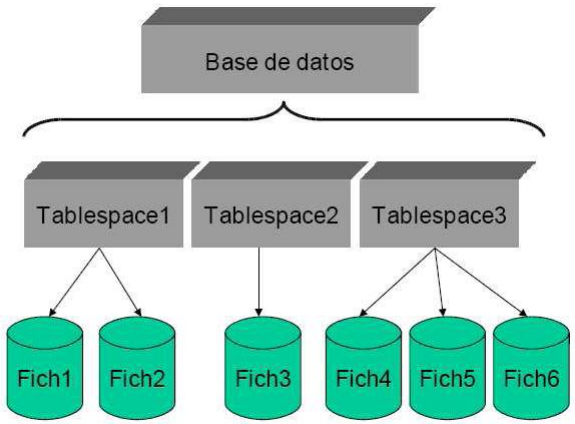
- Appartient à une seule base de données
- Chaque tablespace comprend un ou plusieurs fichiers SE
- Les tablespaces peuvent être mis on/off line lors de l'exécution de la base de données
  - excepté le tablespace SYSTEM
  - et un tablespace avec un rollback segment actif
- Possibilité de basculer entre le statut Read/Write et le statut Read seul.

Fichiers de données

105

- Informations quantitatives :
  - nombre maximum de fichier par tablespace = 1023
  - nombre maximum de tablespace par base de données : 64 000

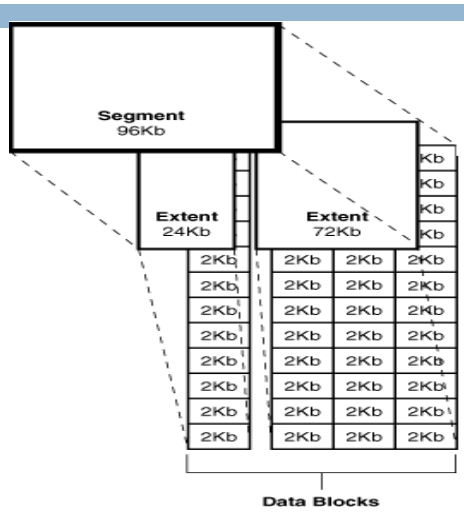
106



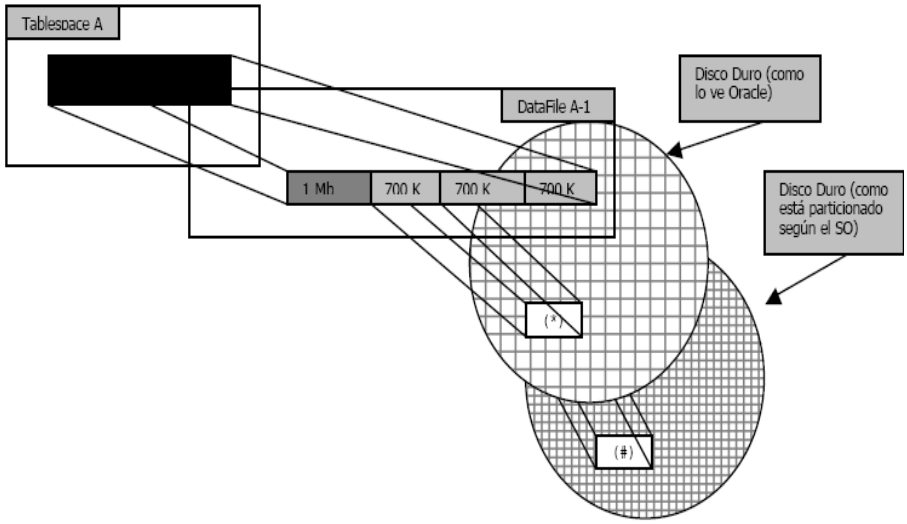
Lien segments/extents/blocs

- Segments
  - Espace alloué à un type spécifique de structure logique de stockage dans un tablespace
    - segments de table, segments d'index, segment temporaire, rollback segment,…
- Extents
  - Ensemble de blocs contigus.
  - Chaque type de segment est composé d'un ou plusieurs extents
- Blocs de données
  - contient un ou plusieurs blocs de fichier physique alloué à partir d'un fichier de données existant
  - plus petite unité d'E/S
  - sa taille vaut DB_BLOC_SIZE

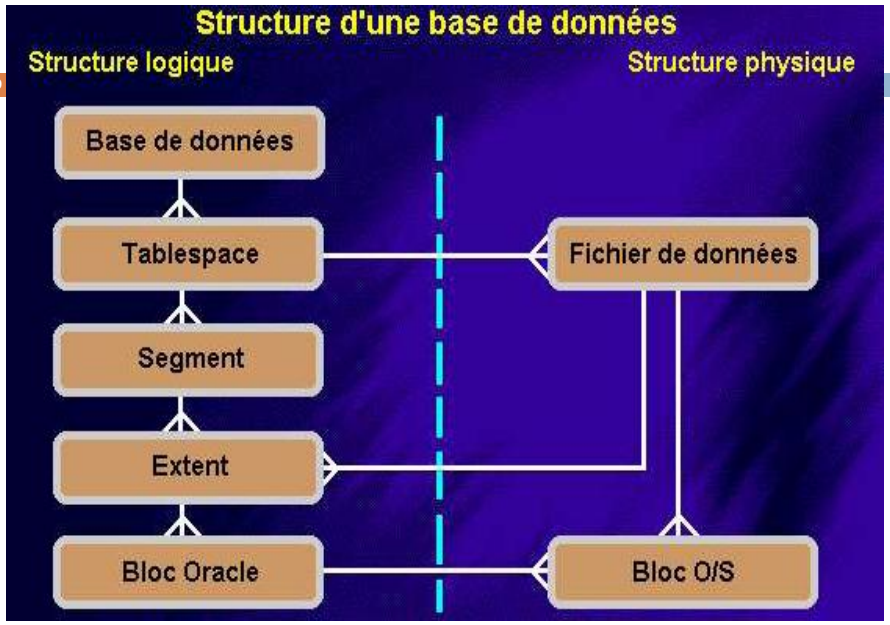
Segments,Extents & bloc (suite)



109


Tablespace A
DataFile A-1
Disco Duro (como lo ve Oracle)
1 Mb
700 K
700 K
700 K
Disco Duro (como está particionado según el SO)
(*)
(#)

110


Structure d'une base de données
Structure logique
Structure physique
Base de données
Tablespace
Fichier de données
Segment
Extent
Bloc Oracle
Bloc O/S

Les Tablespaces

111

- Le tablespace System:
  - Indispensable au fonctionnement de la base de donnée.
  - Il contient les informations du dictionnaire de données, les définitions des procédures stockées, des packages … .
  - Ne devrait pas contenir de données utilisateurs
- Les tablespaces non system:
  - Permettent plus de flexibilité dans l'administration de la BD.
  - Composées de segments de données, d'indexes, de rollback et de segments temporaires.

Création d'un tablespace

112

- Un tablespace est créé à l'aide de la commande
  - Create tablespace

```
Create tablespace tbs_user
Datafile 'c:/oracle/oradata/userdata01.dbf' size 100M
Autoextend on next 5M maxsize 200M;
```

Création d'un tablespace

113

- Syntaxe de création d'un tablespace :

```
CREATE TABLESPACE nom
DATAFILE spécification_fichier_data [,_]
[MINIMUM EXTENT valeur [K|M]]
[ONLINE|OFFLINE]
[PERMANENT|TEMPORARY]
[LOGGING|NOLOGGING]
[EXTENT MANAGEMENT
      DICTIONARY
|LOCAL [AUTOALLOCATE |UNIFORM [SIZE  valeur [K|M]]]]
[DEFAULT clause _ storage ]
[BLOCKSIZE valeur [K]]
```

Création d'un tablespace

114

- DATAFILE: spécifie le ou les fichiers de données liés au tablespace. M spécifie la taille en Mo ou Kbits.
- MINIMUM EXTENT: garantit que la taille de chaque extent du tablespace est un multiple de *valeur*
- LOGGING: option par défaut qui spécifie que toutes les tables, indexes et partitions dans le tablespace génèrent des infos de redo log.
- DEFAULT: définit les paramètres de stockage par défaut pour tous les objets créés dans le tablespace.
- OFFLINE: rend le tablespace indisponible immédiatement après la création.
- PERMANENT: spécifie que le tablespace peut être utilisé pour contenir des objets permanents.
- TEMPORARY: cette clause est utilisée pour les objets temporaires.

Création d'un tablespace

115

- MANAGEMENT DICTIONARY: le tablespace est géré avec le dictionnaire de données.
- MANAGEMENT LOCAL: le tablespace est géré localement et une partie de son espace est réservée pour sa gestion.
  - A la différence des tablespaces standards géré au niveau du dictionnaire de données, la gestion de l'espace physique (allocation / libération de blocs) se fait dans l'entête du fichier(s) du tablespace. Une table binaire d'allocation (bitmap) y est maintenue.
  - Avantages:
    - pas de contention en mise a jour au niveau du dictionnaire
    - …
- AUTOALLOCATE: la taille des extents est gérée par oracle, c'est l'option par défaut
- UNIFORM: il est défini une taille uniforme des extents en k octets ou en M octets . La taille par défaut est de 1 méga Octet

Création d'un tablespace

116

Exemple 1 (tablesapace géré par le dictionnaire de donnée)

```
Create tablespace data
Datafile
  'd:\oracle\oradata\iges5\data01.dbf' size 30M,
  'e:\oracle\oradata\iges5\data02.dbf' size 20M
   autoextend on next 5M maxsize 60M
Extent management dictionary
Minimum extent 150K
Default storage (initial 300K
                 next 300K
                 pctincrease 0
                 minextents 1
                 maxentents 50)
Online;
```

Création d'un tablespace

117

Exemple 2 (tablesapace géré localement avec des extents uniform)

Create tablespace indx
Datafile
'd:\oracle\oradata\iges5\indx01.dbf' size 100M,
Extent management local uniform size 128k;

Création d'un tablespace

118

Exemple 3

Create temporary tablespace temp
TEMFILE 'd:\oracle\oradata\iges5\temp01.dbf' size 100M
autoextend on next 10M maxsize 1000M
extent management local uniform size 4M;

Tablespaces temporaires

119

- Tablespaces temporaires
  - Utilisé pour les opérations de tri
  - Ils ne peuvent pas contenir d'objets permanents
  - La gestion local des exents est recommandée
  - Exemple :

```
CREATE  TEMPORARY TABLESPACE temp
TEMPFILE '/DISK2/temp01.dbf' SIZE 500M
Extent management local uniform size 4M;
```

Les Undo Tablespaces

120

- Les tablespaces de undo (tablespaces d'annulation) sont une nouveauté de Oracle 9i.
  - Ils sont utilisés uniquement pour stocker les segments de undo (annulation)
  - Il ne peut contenir aucun autre objet
  - Il ne peut être utilisé qu'avec la clause datafile

```
CREATE UNDO TABLESPACE undo1
DATAFILE 'u01/oradate/undo101.dbf' SIZE 40M;
```

- Les informations d'utilisation des segments de UNDO sont stockées dans la vue V$UNDOSTAT.

Augmenter la capacité de stockage

121

- Augumenter la taille d'un tablespace
  - Ajouter des fichiers de données au tablespace

    ALTER TABLESPACE data
    ADD DATAFILE
    'd:\oracle\oradata\iges5\data03.dbf' SIZE 200M;
  - Automatiquement avec l'activation de AUTOEXTEND

    ALTER TABLESPACE data
    ADD DATAFILE
    'd:\oracle\oradata\iges5\data02.dbf' SIZE 200M
    AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
- Redimensionner manuellement un fichier de données

  ALTER DATABASE
  DATAFILE 'd:\oracle\oradata\iges5\data02.dbf' RESIZE 60M;

Statut OFFLINE

122

- Les données d'un tablespace offline ne sont plus accessibles
  - L'administrateur peut sauvegarder cette partie des données sans arrêter le fonctionnement du reste de la base
  - Ne peuvent jamais être mis offline
    - le tablespace SYSTEM
    - tout tablespace avec des rollbacks segments actifs
- Exemple :

  ALTER TABLESPACE data OFFLINE;

  Pour rendre de nouveau le tablespace online:
  (ALTER TABLESPACE data ONLINE;)

????Déplacement de fichiers de données

123

- Fermer la base de données (Shutdown)
- Utiliser les commande de système d'exploitation pour déplacer le fichier de données
- Monter la base (Mount)
- Exécuter la commande ALTER TABLESPACE RENAME :

```
ALTER TABLESPACE data
RENAME DATAFILE
'd:\oracle\oradata\iges5\data02.dbf ' TO
'd:\data\DISK5\data02.dbf';
```

- Ouvrir la base

Lecture seule

124

- Empêche toute opération d'écriture sur les fichiers de données
- Permet de faciliter l'administration d'un sous-ensemble stable des données de l'application
  - données peuvent résider sur CDROM
  - Exemple :

```
ALTER TABLESPACE data
READ ONLY;
```

  - Pour rendre de nouveau accessible en modification de données:

```
ALTER TABLESPACE data
READ WRITE;
```

Suppression d'un tablespace

125

- Un tablespace ne peut être supprimé:
  - S'il s'agit du tablespace SYSTEM
  - S'il possède des segments actifs
- Syntaxe :

  DROP TABLESPACE nom
  [INCLUDING CONTENTS
  [CASCADE CONSTRAINTS]] ;
  - INCLUDING CONTENTS : supprime tous les objets (segments) du tablespace
  - CASCADE CONSTRAINTS : supprime les contraintes d'intégrité référentielle des tables définies dans un autre tablespace qui référencent une table du tablespace à supprimer
- Un tablespace contenant encore des données ne peut être supprimé sans l'option INCLUDING CONTENTS
- Ne supprime que les pointeurs de fichiers du fichier de contrôle => les fichiers de données existent toujours
- Pour supprimer les fichiers de données associés à un tablespace il faut ajouter AND DATAFILES

Suppression d'un tablespace

126

- Exemple

  DROP TABLESPACE data
  INCLUDING CONTENTS AND DATAFILES
  CASCADE CONSTRAINTS;

Informations sur les tablespaces

127

- Requêtes sur la vue DBA_TABLESPACES
  - TABLESPACE_NAME
  - NEXT_EXTENT
  - MAX_EXTENTS
  - PCT_INCREASE
  - MIN_EXTLEN
  - STATUS
  - CONTENTS
- Informations sur les fichiers de données v$datafile

  Requêtes sur la vue DBA_DATA_FILES
  - FILE_NAME
  - TABLESPACE_NAME
  - BYTES
  - AUTOEXTENSIBLE
  - MAXBYTES
  - INCREMENT_BY

Informations sur les tablespaces

128

- Information sur les fichiers temporaires:
  - Dba_temp_files
  - V$tempfile
- Informations sur les fichiers de données et les tablespaces à partir du fichier de contrôle
  - Jointure entre V$DATAFILE et V$TABLESPACE sur l'attribut TS#

Quelques conseils

- Séparez les données utilisateurs des données du dictionnaire
- Séparez les données utilisateurs suivant les applications qui les utilisent => notion de charge
  - Afin de diminuer la contention E/S, placer les fichiers des tablespaces différents sur des disques distincts
- Créer tablespace UNDO

GESTION D'UNE INSTANCE ORACLE

Objectifs

140

- Créer et gérer des fichiers de paramètres d'initialisation.
- Démarrer et arrêter une instance.
- Modifier le mode de la Base de Données.
- Restreindre les connexions.
- Surveiller et utiliser des fichiers de diagnostic.

141

- Les utilisateurs administrateurs de la base de données sont responsables de la gestion et de l'administration du serveur Oracle. Des privilèges particuliers sont requis pour permettre de faire ces tâches.
- Deux comptes utilisateur DBA son créés automatiquement et ont le rôle DBA :

142

- **Le compte SYS : Créé lors de l'installation d'Oracle avec tous les privilèges systèmes.**
  - Mot de passe par défaut : change_on_install, à changer après l'installation.

143

- **Le compte SYSTEM : Créé lors de l'installation d'Oracle avec tous les privilèges systèmes.**
  - Mot de passe par défaut : manager à changer après l'installation.

Fichiers de paramètres d'initialisation

144

- Pour démarrer une instance, le serveur oracle doit lire le fichier de paramètres d'initialisation.
  - Connect / as sysdba
  - Startup
- Existe deux types de fichiers de paramètres d'initialisation:
  1. Fichier de paramètre statique, PFILE, généralement nommé initSID.ora
  2. Fichier de paramètres persistant, SPFILE, généralement nommé spfileSID.ora

Fichiers de paramètres d'initialisation

145

- Une instance peut présenter plusieurs fichiers de paramètres d'initialisation
  - Pour optimiser les performances dans certaines situations.

Contenu des fichiers de paramètres d'initialisation

- Les paramètres les plus intéressants sont :

```
instance_name=IGES5
db_name=IGES5
control_files=("D:\oracle\oradata\GI4\CONTROL01.CTL",
   "D:\oracle\oradata\GI4\CONTROL02.CTL",
   "D:\oracle\oradata\GI4\CONTROL03.CTL")
db_block_size=8192
db_cache_size=175112192
java_pool_size=20971520
shared_pool_size=57671680
```

- Ces paramètres sont traditionnellement stockés dans le fichiers init.ora associé à l'instance(appelé pfile : parameter file).

  $ORACLE_home\admin\DB_NAME\pfile\init.ora

Fichiers PFILE initSID.ora

147

- Il s'agit d'un fichier texte
- Il peut être modifié à l'aide d'un éditeur du Système d'exploitation
- Toute modification est apportée manuellement
- Les modification sont apporté au démarrage suivant

Créer un fichiers PFILE initSID.ora

- Créer ce fichier à partir d'un exemple de fichier init.ora
  - Oracle universel installer installe un exemple de fichier
  - Copiez l'exemple à l'aide des commande appropriée du SE
  - Identifiez-le de façon unique à l'aide d'un SID de base de données
    - Cp init.ora $oracle_home/dbs/initdba01.ora
- Modifiez le fichier InitSID.ora
  - Editez les paramètres
  - Affectez des valeurs qui répondent aux besoins de la BDD

Volatilité des paramètres d'initialisation & spfile

- Si un paramètre était modifié via la commande alter uniquement, au prochain redémarrage de l'instance les changements sont perdus à moins d'avoir modifié manuellement init.ora
  - Alter system set undo_tablesapce='UNDO2'
  - Solution : Fichiers de paramètres serveur : spfile Format binaire avec paramètres d'initilaisation non volatils.
- $ORACLE_HOME\database\spfile_SID.ora

Fichiers SPFILE SpfileSID.ora

150

- Il s'agit d'un fichier binaire
- Sa mise à jour est effectuée par le serveur oracle
- Il réside toujours côté serveur
- Il permet de rendre les modifications persistantes après l'arrêt et le redémarrage

Créer un fichiers SPFILE

151

- Créez un fichier SPFILE à partir d'un fichier PFILE

```
SQL> CREATE SPFILE ='$oracle_home/dbs/SPFILEdb01.ora'
        from PFILE ='$oracle_home/admin/DB_NAME/pfile/initDB01.ora'
```

- Il peut être exécuté avant ou après le démarrage de l'instance (Privilège SYSDBA)

Modifier des paramètres du fichier SPFLE

152

- Utilisez la commande ALTER SYSTEM pour apporter des modifications aux valeurs de paramètres:
  - Alter system set undo_tablesapce='UNDO2'
  - Indiquez si ces modifications sont temporaires ou persistantes:
    - Alter system set undo_tablespace='UNDO2'

      scope=BOTH
      - Scope=memory|spfile|both
        - Memory : modifier la valeur du paramètre uniquement dans l'instance en cours
        - Spfile: modifier la valeur du paramètre uniquement dans l'instance en cours
        - Both :modifier la valeur du paramètre dans l'instance en cours et le spfile

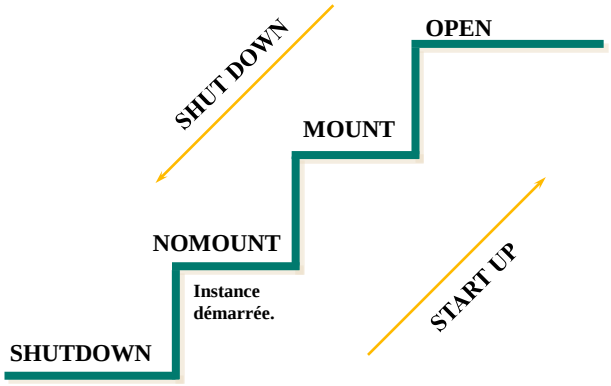
Fonctionnement de la commande STARTUP

153

- Ordre de priorités:
  - spfileSID.ora
  - SPFILE par défaut
  - initSID.ora
  - PFILE par défaut
- Vous pouvez modifier ces priorités si vous indiquez:
  - Startup pfile= $oracle_home/admin/DB_NAME/pfile/initDB01.ora

Démarrer une base de données en mode NOMOUNT

154



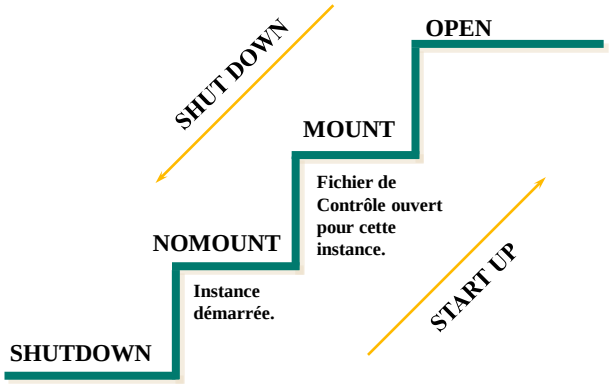
Démarrer une base de données en mode NOMOUNT

155

- Le démarrage d'une instance en mode NOMOUNT ne s'effectue qu'à la création de la BDD ou la récréation de fichiers de contrôle
- Le démarrage en mode NOMOUNT comprend les tâches suivantes:
  - Lecture du fichier d'initialisation dans l'ordre de priorité (spfile/pfile)
  - Allocation des zones mémoires
  - démarrage des processus d'arrière plan
  - L'ouverture du fichier alertSID.log et des fichiers de trace

Démarrer une base de données en mode MOUNT

156



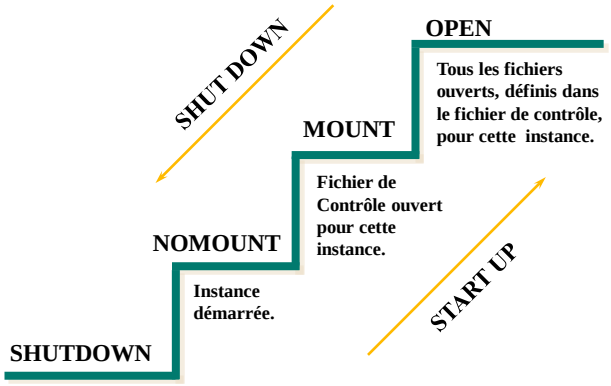
Démarrer une base de données en mode MOUNT

157

- Démarrage de l'instance et lecture des fichiers de contrôle
- Localiser les fichiers de données sans les ouvrir(infos dans les fichiers de contrôle)
- Monter la base de données pour effectuer des opérations de maintenance:
  - Renommer des fichiers de données
  - Activer ou désactiver des options d'archivage de fichiers de journalisation
  - Effectuer un récupération complète de la base de données

Démarrer une base de données en mode OPEN

158



Démarrer une base de données en mode OPEN

159

- Ouvrir la base de données en mode de fonctionnement normale
  - Les utilisateurs sont autorisés peuvent se connecter à la BDD et effectuer les opérations standard sur la base de données.
  - L'ouverture de la BDD comprend les tâches suivantes:
    - Ouverture les fichiers de données online
    - Ouverture des fichiers de journalisation en ligne
  - Au cours de cette dernière étape, le serveur oracle vérifie que tous les fichiers de données et de journalisation en ligne peuvent être ouverts et contrôle la cohérence de la BDD.
  - Si un fichier de données ou de journalisation en ligne manque, le serveur oracle renvoie une erreur.

La commande STARTUP

160

- Démarrez l'instance et ouvrez la BDD
- Vous pouvez modifier ces priorités si vous indiquez:
  - Startup pfile= $oracle_home/admin/DB_NAME/pfile/initDB01.ora

```
Startup [FORCE] [RESTRICT] [PFILE=name]
        [OPEN [RECOVER] [DATABASE]
        |MOUNT
        NOMONT]
```

Force: interrompt l'instance en cours, puis exécute un démarrage normale

Restrict : n'autorise l'accès à la BDD qu'aux utilisateurs disposant du privilège RESTRICTED SESSION

Recover: lance la procédure de la restauration au démarrage physique la BDD

La commande STARTUP

161

- Dépannage:
  - Si vous rencontrez des erreurs à l'exécution de la commande startup. Vous devez d'abort lancer la commande shutdown AVANT DE REEXECUTER LA COMMANDE STARTUP
- Remarque:
  - Startup et Shutdown son des commande de SQL*PLUS et non SQL

Commande ALTER DATABASE

- Remplacez le status NOMOUNT de la BDD par le status MOUNT:
  - ALTER DATABASE name MOUNT;
- Du status MONT à OPEN

  ALTER DATABASE name OPEN;

- Ouvrez la base de données en lecture seule:
  - ALTER DATABASE name READ ONLY;

- Mode lecture et écriture

  ALTER DATABASE name READ WRITE;

Ouvrir une BDD en mode d'accès restreint

- Utilisez la commande STARTUP pour restreindre l'accès à une BDD
  - STARTUP RESTRICT
- Utilisez la commande ALTER SYSTEM pour placer une instance en mode d'accès restreint:
  - ALTER SYSTEM ENABLED RESTRICTED SESSION;
- Utilisez la commande ALTER SYSTEM pour désativer le mode d'accès restreint:
  - ALTER SYSTEM DISABLE RESTRICTED SESSION;

Ouvrir une BDD en mode d'accès restreint

164

- Pour mettre fin à une session
  - ALTER SYSTEM KILL SESSION 'integer1,integer2'
    - Ou integer1 = SID de la Vue v$session et integer2 le SERIAL# de la même Vue.
- À l'exécution de ALTER SYSTEM KILL SESSION, le processus PMON effectue les tâches suivantes:
  - Annulation de la transaction en cours de l'utilisation
  - Libration de tous les verrous de table ou de ligne
  - Libération de tous les ressources réservées par l'utilisateur

Ouvrir une BDD en mode lecture seule

165

- Ouvrir une BDD en mode lecture seule
  - STARTUP MOUNT

    ALTER DATABASE OPEN READ ONLY
- Une BDD en lecture seule permet:
  - D'exécuter des interrogations
  - D'exécuter des tris sur disque à l'aide de tablespaces gérés localement,
  - De mettre des fichiers de données hors ligne et en ligne, mais pas des tablespaces,
  - De récupérer des fichiers de données et des tablespaces hors lignes;

Arrêt de la base de données

166

- Arrêtez la BDD pour :
  - effectuer la sauvegarde hors ligne de toutes les structure physique via le système d'exploitation
  - Appliquer les modifications aux paramètres d'initialisation statique

Arrêt de la base de données

167

- Mode d'arrêt:
  - ABORT (A)
  - IMMEDIATE (I)
  - TRANSACTIONAL (T)
  - NORMAL (N) (mode par défaut)

Mode d'arrêt	A	I	T	N
Permet de nouvelles connexions	Non	Non	Non	Non
Attend la fin des sessions en cours	Non	Non	Non	Oui
Attend la fin des transaction en cours	Non	Non	Oui	Oui
Applique un point de reprise et ferme les fichiers	Non	Oui	Oui	Oui

Arrêt en mode Normal, Transactional ou immediate

- Phases d'arrêt:
  1. Le cache de tampon de la BDD est écrit dans les fichiers de données,
  2. Les modification non validées sont annulées,
  3. Les ressources sont libérées

     (base de données cohérente (base « propre »)
  4. Aucune récupération d'instance

Arrêt en mode Normal

- Arrêt en mode Normal
  - Aucune nouvelle connexion ne peut être établie
  - Le serveur oracle attend la déconnexion préalable de tous les utilisateurs
  - Les tampon de journalisation et de données sont écrits sur disque
  - Oracle ferme et démonte la base de données
  - Les processus d'arrière plan prennent fin et la zone SGA est supprimée de la mémoire
  - La récupération de l'instance n'est pas nécessaire lors du redémarrage

Arrêt en mode Transactional

170

- Arrêt en mode transactionnel (évite aux client de perdre leurs travaux en cours)
  - Aucune nouvelle connexion ne peut être établie
  - Le client est déconnecté lorsqu'il termine la transaction en cours
  - La fin de toutes les transactions entraîne l'arrêt immédiat de la BDD
  - La récupération de l'instance n'est pas nécessaire lors du redémarrage

Arrêt en mode Transactional

171

- Arrêt en mode immediate
  - Aucune nouvelle connexion ne peut être établie
  - Les instructions SQL en cours de traitement par oracle ne sont pas terminées
  - Le serveur oracle n'attend pas la déconnexion des utilisateur de la BDD
  - Oracle annule les transaction actives et déconnecte tous les utilisateurs
  - Oracle ferme et démonte la BDD après il arrête l'instance
  - La récupération de l'instance n'est pas nécessaire lors du redémarrage

Arrêt en mode ABORT

- Si les arrêt en modes normale et immediate échouent, vous pouvez abandonner l'instance de la BDD en cours avec le mode ABORT:
  1. Les instructions SQL en cours de traitement par oracle ne sont immédiatement interrompues
  2. Le serveur oracle n'attend pas la déconnexion des utilisateur de la BDD
  3. Les tampons de la BDD et de journalisation ne sont pas écrit dans les fichiers de données,
  4. La base de données n'est pas fermée ni démontée
  5. Une récupération d'instance est nécessaire au démarrage (c'est automatique)
- Déconseiller de sauvegarder une base de données incohérente.

Les Objets d'une Base de données Oracle

Les Objets d'une Base de données Oracle

- Tables
- Vues
- Séquences
- Index
- Synonymes

Les Types de données

Type de données	Description
VARCHAR2 (taille)	Texte de longueur variable. La longueur minimale est de 1 caractère et maximale 4000 caractères. La taille devra obligatoirement être définie.
CHAR (taille)	Texte de longueur fixes. La taille minimum de 1 caractère et pouvant aller jusqu'à 2000 caractères.
NUMBER (p, e)	Nombre ayant une précision pouvant aller de 1 à 38 chiffres et ayant une échelle pouvant aller de –84 à 127. (L'échelle est en fait le nombre de chiffre à afficher après la virgule.)
DATE	Date notée sous la forme DD-MON-YY.

Les Types de données

176

Type de données	Description
LONG	Texte de longueur variable pouvant stocker jusqu'à 2 gigas. On ne pourra mettre qu'une colonne de type LONG par table.
RAW (taille)	Equivalent à VARCHAR2, mais il permet de stocker des données binaires qui ne sont pas interprétées par Oracle. La taille maximum est de 2000.
LONGRAW	Equivalent à LONG mais pour des données de type binaire non interprétées par Oracle.

Les Types de données

177

Type de données	Description
CLOB	Permet de stocker un pointeur vers un fichier de données composé de caractère et pouvant contenir jusqu'à 4 gigas.
BLOB	Permet de stocker un pointeur vers un fichier composé de données binaire et pouvant contenir jusqu'à 4 gigas.
BFILE	Permet de stocker les données binaires d'un fichier externe pouvant contenir jusqu'à 4 gigas.

Les Types de données

178

TABLE 3.1 Precision, Scale, and Rounding

Specification	Actual Value	Stored Value
NUMBER(11,4)	12345.6789	12345.6789
NUMBER(11,2)	12345.6789	12345.68
NUMBER(11,-2)	12345.6789	12300
NUMBER(5,2)	12345.6789	Error – Precision is too small
NUMBER(5,2)	123456	Error – Precision is too small

Tables

179

- Est une unité de stockage élémentaire, composée de lignes et de colonnes
- Le nom d'une table dans une BDD Oracle
  - Doit commencer par une lettre
  - Ne doit pas dépasser les 30 caractères
  - Ne peut contenir que les caractères A à Z, a à z, 0 à 9, _, $, et #
  - Ne doit pas porter le nom d'un autre objet appartenant au même utilisateur
  - Ne doit pas être un mot réservé

Tables

180

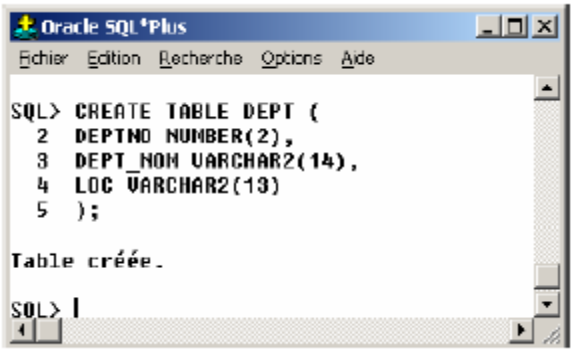
- Pour créer une table, vous devez avoir le privilège CREATE TABLE

Syntaxe

```
SQL> CREATE TABLE [Schema.]nom_table
        (colonne1 datatype [Default expr] [Contraintes_Colonne] [,
                  colonne2 datatype [Default expr] [Contraintes_Colonne],
                  … ]
        [Contraintes_Table]
);
```

Tables

181



Tables

182

```
CREATE TABLE EMP2
  (EMPNO NUMBER(2),
  ENAME VARCHAR2(50),
  LOCATION VARCHAR2(50))
  TABLESPACE DATA1;
```

Tables

183

- Attention
  - Pour accéder à une table d'un autre utilisateur, il faut précéder le nom de la table par le nom du propriétaire
    - Ex: SCOTT.EMP
  - La création d'une table fait appel à un ordre du LDD

    ⇒sa validation est automatique
- L'option DEFAULT
  - Permet de définir la valeur par défaut
    - Ex:
      - DATE_PRET DATE DEFAULT SYSDATE
      - DEPARTEMENT VARCHAR2(25) DEFAULT 'Commercial'

Création des tables en utilisant une sous-requête

184

- Il est possible de créer une table et insérer des lignes en utilisant la requête **CREATE TABLE et** l'option ***AS sous-requête.***

CREATE TABLE ***table***
[(colonne, colonne …)]
AS ***sous-requête ;***

Création des tables en utilisant une sous-requête

185

- Exemple

SQL> CREATE TABLE copy_emp AS SELECT * FROM emp;

- Explication : On créée une nouvelle table *copy_emp qui possède la même structure et les mêmes* données que la table *emp*.

Création des tables en utilisant une sous-requête

186

- Exemple

```
SQL> CREATE TABLE copy_emp
        NOLOGGING
        TABLESPACE DATA2
       AS SELECT * FROM emp
          WHERE EMPNO<20;
```

Création des tables en utilisant une sous-requête

187

- ATTENTION
  - Le nombre de colonnes spécifiées doit correspondre au nombre des colonnes de la requête
  - La définition d'une colonne ne peut contenir que son nom et sa valeur par défaut
  - Si aucune colonne n'est spécifiée, la table aura les mêmes colonnes que celles de la requête
  - Les contraintes ne sont pas prises en compte

Modification des tables

188

- L'ordre ALTER TABLE
  - Permet de modifier la structure d'une table (Ajout d'une colonne, modification, définition d'une valeur par défaut, supprimer une colonne, etc.)
  - La modification d'une valeur par défaut ne s'appliquer qu'aux insertions ultérieures dans la table

Modification des tables

189

- L'ordre ALTER TABLE
  - Apres avoir crée une table, il est parfois nécessaire de modifier sa structure. Pour modifier une table il faudra employer la commande : **ALTER TABLE nom_de_la_table**

ALTER TABLE ***tablename***
ADD | MODIFY | DROP ***(column datatype [DEFAULT expr]);***

Modification des tables

190

- L'ordre ALTER TABLE

```
ALTER TABLE EMPLOYEE
ADD  TELEPHONE VARCHAR2(10);

ALTER TABLE EMPLOYEE
ADD  (fax VARCHAR2(10)
        email VARCHAR2(50));
```

Modification des tables

191

- L'ordre ALTER TABLE

```
ALTER TABLE EMPLOYEE
MODIFY  TELEPHONE VARCHAR2(20);
```

Modification des tables

192

- L'ordre ALTER TABLE

```
ALTER TABLE EMPLOYEE
DROP COLUMN  TELEPHONE
CASCADE CONSTRAINTS;
```

Modification des tables

194

- L'ordre RENAME
  - Permet de modifier le nom d'une table, d'une vue, d'une séquence, et d'un synonyme
- Syntaxe

SQL> RENAME nom_objet TO nouveau_Nom;

Ou

SQL> ALTER TABLE nom_table RENAME TO nouveau_Nom;

  - Vous devez être propriétaire de l'objet

Supprimer une table

195

- L'ordre DROP TABLE
  - Syntaxe

    SQL> DROP TABLE [schema.]nom_table
    CASCADE CONSTRAINTS;

    - **La clé étrangère doit être supprimée avant la table parent (cascade constraints)**

Supprimer une table

196

- L'ordre DROP TABLE

  SQL> DROP TABLE emp;

- Pour supprimer une table, l'utilisateur Doit être le créateur
- Il possède le privilège DROP ANY TABLE

Vider une table

197

- L'ordre TRUNCATE TABLE
  - Permet de vider une table et libérer l'espace de stockage utilisé par la table
- Syntaxe

  SQL> TRUNCATE TABLE nom_table;
- Cet ordre est irréversible
- Vous pouvez utiliser l'ordre DELETE pour supprimer les lignes d'une table
  - Mais il ne libère pas l'espace de stockage

Tables

198

- L'ordre COMMENT ON
  - Permet d'ajouter des commentaire à une table
- Syntaxe

  SQL> COMMENT ON [TABLE nom_table | COLUMN nom_table.nom_colonne]IS 'commentaire';

Tables

199

- L'ordre COMMENT ON

```
COMMENT ON TABLE EMP IS
'Employes';

COMMENT ON COLUMN EMP.TEL IS
 'Numero de Telephone';
```

Tables

200

- L'ordre COMMENT ON
  - Pour afficher les commentaire, on peut utiliser
    - ALL_COL_COMMENTS
    - USER_COL_COMMENTS
    - ALL_TAB_COMMENTS
    - USER_TAB_COMMENTS

Tables

201

- Intégrité de données
  - Garantit que les données d'une base respectent certaines règles
    - pour empêcher l'utilisateur d'entrer des données invalides dans la base
  - Empêchent la suppression d'une table lorsqu'il existe des dépendances

tables

202

- Toutes les contraintes sont stockées dans le dictionnaire de données (dans la table USER_CONSTRAINTS).
  - Sont nommées sous Oracle
    - Automatiquement sous le format SYS_Cn
    - Manuellement par l'utilisateur
      - Lors de la création (CREATE TABLE)
      - Après la création (ALTER TABLE)
  - Donner des noms explicites à vos contraintes pour faciliter le référencement.
- Les contraintes peuvent être définies soit au moment de la création de la table soit après la création.

Tables

203

- Types de contraintes valides

NOT NULL	**Spécifie que cette colonne ne doit pas contenir une valeur NULL**
UNIQUE	**Spécifie une colonne (ou une combinaison de colonnes) dont les valeurs doivent être uniques**
PRIMARY KEY	**Identifie la clé primaire**
FOREIGN KEY	**Identifie la clé étrangère**
CHECK	**Spécifie une condition à laquelle doit répondre chaque ligne de la table**

Tables

204

- **La contrainte NOT NULL**

```
CREATE TABLE employee (
  employee_id NUMBER(6),
  nom VARCHAR2(25) NOT NULL, ➔NOMME PAR LE SYSTEME
  sal NUMBER(8,2),
  hiredate DATE CONSTRAINT emp_hiredate_nn NOT NULL,
  ➔NOMME PAR L'utilisateur …);
```

Tables

205

- Les contraintes
  - PRIMARY KEY
    - Définit :

Au niveau table

```
SQL> CREATE TABLE EMP (
         empno NUMBER(4),
         ename VARCHAR2(10),
         …
         deptno NUMBER(7,2) NOT NULL,
         CONSTRAINT emp_emmno_pk PRIMARY KEY (EMPNO)
);
```

Au niveau colonne

```
SQL> CREATE TABLE EMP (
         empno NUMBER(4) PRIMARY KEY,
         ename VARCHAR2(10),
         …
         deptno NUMBER(7,2) NOT NULL,
         );
```

Tables

206

Les contraintes

- FOREIGN KEY
  - Définit au niveau table ou colonne
  - Associé à
    - REFERENCES pour identifier la table et la colonne de la table maître
    - ON DELETE CASCADE pour autoriser la suppression d'une ligne dans la table maître et des lignes dépendantes dans la table détail

Tables

207

- Les contraintes
  - FOREIGN KEY
    - Exemple

Au niveau Table

```
SQL> CREATE TABLE EMP (
          empno NUMBER(4) PRIMARY KEY,
          ename VARCHAR2(10),
          …
          deptno NUMBER(7,2) NOT NULL,
          CONSTRAINT emp_deptno_KT FORRIGN KEY (deptno) REFERENCES DEPT (deptno));
```

Au niveau Colonne

```
SQL> CREATE TABLE EMP (
          empno NUMBER(4) PRIMARY KEY,
          ename VARCHAR2(10),
          …
          deptno NUMBER(7,2) NOT NULL REFERENCES DEPT (deptno));
```

Tables

208

- Les contraintes
  - CHECK

```
SQL> CREATE TABLE EMP (
          empno NUMBER(4) PRIMARY KEY,
          ename VARCHAR2(10),
          …
          deptno NUMBER(7,2) NOT NULL,
          CONSTRAINT emp_deptno_KT CHECK (deptno BETWEEN 10 AND 99), …));
```

Tables

209

Les contraintes

- UNIQUE
  - Définit au niveau table ou colonne

Au niveau table

```
SQL> CREATE TABLE EMP (
          empno NUMBER(4) PRIMARY KEY,
          ename VARCHAR2(10),
          …
          deptno NUMBER(7,2) NOT NULL,
          CONSTRAINT emp_nom_KT UNIQUE (ename)
);
```

Au niveau Colonne

```
SQL> CREATE TABLE EMP (
          empno NUMBER(4) PRIMARY KEY,
          ename VARCHAR2(10) UNIQUE,
          …
          deptno NUMBER(7,2) NOT NULL,
);
```

Tables

210

- Ajout d'une contrainte

  ALTER TABLE table

  ADD [CONSTRAINT constriant] type (column);

- Pour ajouter la contrainte NOT NULL utilisez la clause MODIFY

  ALTER TABLE table

  MODIFY column NOT NULL;

Tables

211

Les contraintes

- Suppression d'une contrainte

```
SQL> ALTER TABLE nom_table
DROP PRIMARY KEY | UNIQUE (colonne) |
          CONSTRAINT nom_contrainte [CASCADE]
);
```

- Ex: Supprimer la contrainte PRIMARY KEY de la table DEPT, ainsi que la contrainte FOREIGN KEY définie sur la colonne DEPTNO de la table EMP

```
SQL> ALTER TABLE DEPT (
          DROP PRIMARY KEY CASCADE
);
```

Tables

212

Les contraintes

- Suppression d'une contrainte

ALTER TABLE EMP
DROP CONSTRATINT cn_nom;

Tables

213

Les contraintes

- Désactivation/Activation d'une contrainte

```
SQL> ALTER TABLE nom_table
[DISABLE | ENABLE] CONSTRAINT nom_contrainte [CASCADE]
);
```

☺ On peut utiliser les clauses ENABLE et DISABLE dans CREATE TABLE et ALTER TABLE

Tables

214

- Désactivation de contraintes

```
ALTER TABLE   Emp
 DISABLE CONSTRAINT pk_emp CASCADE;
```

```
ALTER TABLE   Emp
 DISABLE CONSTRAINT cn_nom;
```

Tables

215

- Activation de contraintes

```
ALTER TABLE   Emp
 ENABLE CONSTRAINT pk_emp;
```

Tables

216

Les contraintes

- Comment afficher les contraintes sur une table

```
SQL> SELECT constraint_name, constraint_type, search_condition
          FROM user_constraints
          WHERE table_name = 'EMP'
);
```

Constraint_Name	Constraint-type	Search_Condition
SYS_C003042	**C** (Signifie CHECK)	EMPNO is not Null
SYS_C003044	**U** (Signifie Unique)	
TEST_TT	**P** (Signifie Primary Key)	

Tables

217

Les contraintes

- Comment afficher les colonnes associées aux contraintes dans une table

```
SQL> SELECT constraint_name, column_name
          FROM user_cons_column
          WHERE table_name = 'EMP'
);
```

Constraint_Name	Column Name
TEST_TT	EMPNO
SYS_C003042	DEPTNO

Données

218

L'ordre INSERT

- Permet d'ajouter de nouveaux enregistrements dans une table
- Syntaxe

```
SQL> INSERT INTO nom_table [(colonne_1 [, colonne_2…)]
          VALUES (Valeur_colonne_1 [, valeur_colonne_2…]
);
```

- Placez les valeurs de type caractère et date entre simples quotes

Données

219

L'ordre INSERT

- Exemple
  - En précisant les colonnes concernées

```
SQL> INSERT INTO Emp (empno, ename, sal)
          VALUES (5555, 'BARTH', 2500);
```

  - Sans précision

```
SQL> INSERT INTO Emp
          VALUES (5555, 'BARTH', NULL, NULL, SYSDATE, 2500, NULL, NULL);
```

Données

220

L'ordre INSERT

- Exemple
  - Avec des variables saisies par l'utilisateur

```
SQL> INSERT INTO Emp (empno, ename, sal)
          VALUES (&Num_Salarie, '&Nom_salarie', &salaire);
```

Données

221

L'ordre INSERT

- Exemple
  - A partir d'une autre table

```
SQL> INSERT INTO Emp (empno, ename, sal)
              SELECT empno, ename, sal FROM SCOTT.EMP WHERE job = 'manager';
```

Données

222

L'ordre UPDATE

- Permet de modifier les enregistrements existants
  - A partir des valeurs

```
SQL> UPDATE Nom_table
        SET Colonne_1 = valeur_1 [, Colonne_2 = …]
        [WHERE condition] ;
```

  - A partir d'une requête

```
SQL> UPDATE Nom_table
        SET (Colonne_1, Colonne_2, ) =
        (SELECT colonne_11, Colonne_22, …
         FROM nom_table_2
         WHERE condition)
     [WHERE condition] ;
```

Données

223

L'ordre UPDATE

- Exemple
  - Modifier le poste et le n° de département de l'employé 7698 à l'identique de l'employé 7499

```
SQL> UPDATE Emp
         SET (job, deptno) =
         (SELECT job, deptno
          FROM Emp
          WHERE empno = 7499)
     WHERE empno = 7698 ;
```

Données

224

L'ordre DELETE

- Permet de supprimer les enregistrements d'une table

```
SQL> DELETE [FROM] nom_table
     [WHERE condition] ;
```

- ☠ Attention, si vous omettez la clause WHERE, toutes les lignes sont supprimées

Vues

225

- Les vues sont des tables virtuelles qui permettent de définir des filtres sur des tables réelles.
  - Une Vue est requête pouvant être manipulée comme une table
  - Contrairement à une table, une vue stocke aucune donnée, seulement une instruction SQL
- Elles permettent de limiter l'accès à certaines colonnes d'une table pour un groupe d'utilisateurs et le type d'accès

Vues

226

- Elles permettent de simplifier l'accès aux données

Syntaxe

```
SQL> CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW Nom_Vue
          [(alias [, alias]…)]
AS requête
[WITH READ ONLY]
```

REPLACE	Recrée la vue
FORCE	Crée la vue même si les tables n'existent pas
WITH READ ONLY	Garantit qu'aucune opération LMD ne peut être exécutée dans la vue

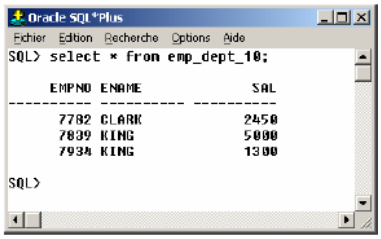
Vues

227

Exemple

- Sans alias

```
SQL> CREATE OR REPLACE VIEW
EMP_DEPT_10
        AS SELECT empno, ename, sal
        FROM Emp
        WHERE deptno = 10;
```



Vues

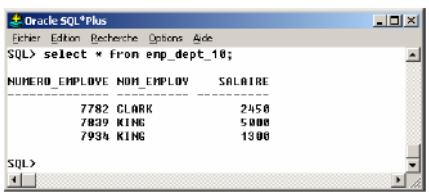
228

Exemple

- Avec des alias

```
SQL> CREATE VIEW EMP_DEPT_10
        AS SELECT empno Numero_Employe, ename Nom_Employe, sal Salaire
        FROM Emp
        WHERE deptno = 10;
```

```
SQL> CREATE VIEW EMP_DEPT_10 (Numero_Employe, Nom_Employe, Salaire)  AS
SELECT empno, ename, sal
        FROM Emp
        WHERE deptno = 10;
```



Vues

229

Règles d'exécution des ordres LMD

- Vous ne pouvez pas
  - Supprimer un enregistrement si la vue contient
    - Des fonctions de groupe
    - Une clause GROUP BY
    - Le mot-clé DISTINCT

Vues

230

Règles d'exécution des ordres LMD

- Vous ne pouvez pas
  - Modifier les données si la vue contient
    - Des fonctions de groupe
    - Une clause GROUP BY
    - Le mot-clé DISTINCT
    - Des colonnes définies par des expressions

Vues

231

Règles d'exécution des ordres LMD

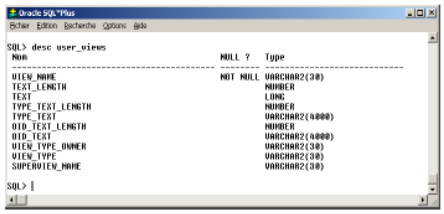
- Vous ne pouvez pas
  - Ajouter de données si
    - Des fonctions de groupe
    - Une clause GROUP BY
    - Le mot-clé DISTINCT
    - Des colonnes définies par des expressions
    - Les tables de base contiennent des colonnes NOT NULL non sélectionnées par la vue
      - Sans valeur par default

Vues

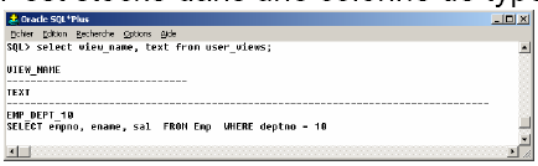
232

La table USER_VIEWS

- Permet d'afficher le nom et la définition des vues de l'utilisateur



- L'ordre SELECT est stocké dans une colonne de type LONG



Vues

233

L'ordre DROP VIEW

- Permet de supprimer une vue
- N'entraîne pas la perte des données
- Exemple

```
SQL> DROP VIEW EMP_DEPT_10;
Vue supprimée.
```

Séquences

234

- Une séquence est un objet créé par l'utilisateur.
- Elle sert à créer des valeurs pour les clés primaires, qui sont incrémentées ou décrémentées par le serveur Oracle.
- Noter que la séquence est stockée et générée indépendamment de la table, et une séquence peut être utilisée pour plusieurs tables.

Séquences

235

Syntaxe

[CREATE | ALTER] SEQUENCE ***[schéma].Nom_séquence***

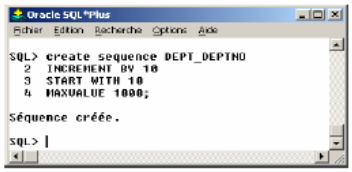
[INCREMENT BY ***entier***]	→ définit l'intervalle d'incrémentation ou de -
[START WITH ***entier***]	→ Première valeur de la séquence
[MAXVALUE ***entier***]	→ Valeur maximale (par défaut 1027 ou -1)
[MINVALUE ***entier***]	→ Valeur minimale (par défaut 1 ou -1026)
[CYCLE | NOCYCLE]	→ Revenir à la valeur initiale
[CACHE ***entier*** | NOCACHE]	→ allocation de séquences en mémoire (par défaut 20)

Séquences

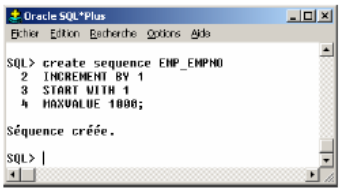
236

Exemple

Créer un compteur DEPT_DEPTNO pour la clé primaire de la table DEPT



Créer un compteur pour la clé primaire de la table EMP

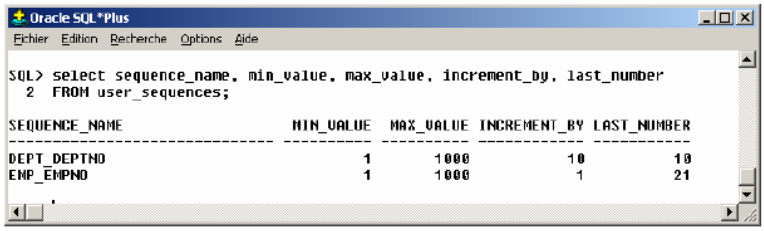


Séquences

237

La table USER_SEQUENCES

- Contient les valeurs des séquences définies par l'utilisateur



Séquences

238

PseudoColonnes

- NEXTVAL
  - Donne la prochaine valeur de la séquence.
  - Retourne une valeur unique même si elle est utilisée par plusieurs utilisateurs
- CURRVAL
  - Renvoie la valeur courante de la séquence utilisée pour chacun des utilisateurs

Séquences

239

Utilisation

- NEXTVAL

```
SQL> INSERT INTO emp (empno, ename, sal)
        VALUES (emp_empno.NEXTVAL, 'rich', 2000);
```

- CURRVAL

```
SQL> SELECT emp_empno.CURRVAL FROM dual;
```

Séquences

240

Utilisation

- Pour utiliser la séquence d'un autre utilisateur, il faut préciser le nom du schéma (schéma.nom_séquence)

```
SQL> SELECT scott.emp_empno.CURRVAL FROM dual;
```

- Il faut avoir
  - les privilèges objet SELECT
  - Les privilèges Système SELECT ANY SEQUENCE

Séquences

241

Règles de modification

- Etre propriétaires ou avoir le privilège Objet ALTER
- ALTER SEQUENCE ne modifie que les numéros de séquence à venir
- Vous devez supprimer la séquence puis la recréer si vous voulez modifier le premier numéro (START WITH)

Séquences

242

L'ordre DROP SEQUENCE

- Permet de supprimer une séquence
- Syntaxe

```
SQL> DROP SEQUENCE Nom_séquence;
```

Synonymes

243

- Un synonyme est un nom alternatif pour désigner un objet de la base de données.
  - C'est aussi un objet de la base de données.
- Si un utilisateur veut accéder à un objet appartenant à un autre schéma, il faut le faire préfixer par le nom de schéma (SCOTT.emp)

Synonymes

244

Définition

- Permet de renommer un objet dans le schéma (Table, Vue, séquence, etc.)
- Elimine la contrainte de désigner l'objet avec son schéma

Syntaxe

```
SQL> CREATE [PUBLIC] SYNONYM nom_synonyme
       FOR objet;
```

Public donne l'accès à tous les utilisateurs

Synonymes

245

Exemple

```
SQL> CREATE PUBLIC SYNONYM ma_sequence
       FOR EMP_EMPNO;
```

Suppression

```
SQL> DROP SYNONYM nom_synonyme;
```

Index

246

- Un index est un objet qui peut augmenter la vitesse de récupération des lignes en utilisant les pointeurs.
- Les index peuvent être créés automatiquement par le serveur Oracle ou manuellement par l'utilisateur.
- Ils sont indépendants donc lorsque vous supprimez ou modifiez un index les tables ne sont pas affectées.

Index

247

- Lors de la création des clés primaires ou des clés étrangères, les index sont crées automatiquement et ils possèdent les même noms que les contraintes.

Index

248

- **Quand créer un index:**
  - La colonne contient une large plage de valeurs.
  - La colonne contient plusieurs valeurs nulles.
  - Une ou plusieurs colonnes sont fréquemment utilisées dans la clause **WHERE** ou pour les conditions de jointure.
  - La table est grande et la plupart des requêtes recherchent moins de 2-4% des lignes.

Index

249

- **Quand ne pas créer des index:**
  - La table est petite.
  - Les colonnes ne sont pas souvent utilisées.
  - Les requêtes recherchent plus de 2-4% des lignes.
  - La table est mise à jour fréquemment.
  - Les colonnes indexées sont référencées comme une partie de l'expression.

Index

250

- Pour créer un index, il faut utiliser la requête **CREATE INDEX**

CREATE INDEX ***nomindex***
ON ***table (column1, column2 …);***

- Exemple :

```
SQL> CREATE INDEX emp_ename_idx
        ON emp(ename);
```

Index

251

- Pour créer un index, il faut utiliser la requête **CREATE INDEX**

  CREATE INDEX ***nomindex***
  ON ***table (column1, column2 …)***
  [TABLESPACE INDX];

- Exemple :

  SQL> CREATE INDEX emp_ename_idx
  ON emp(ename);

Index

252

- Pour vérifier l'existence d'un index, vous pouvez interroger la table USER_INDEXES

- **Supprimer un index**
  - Il n'est pas possible de modifier un index, il faut le supprimer ou recréer.

    DROP INDEX ***nomindex;***

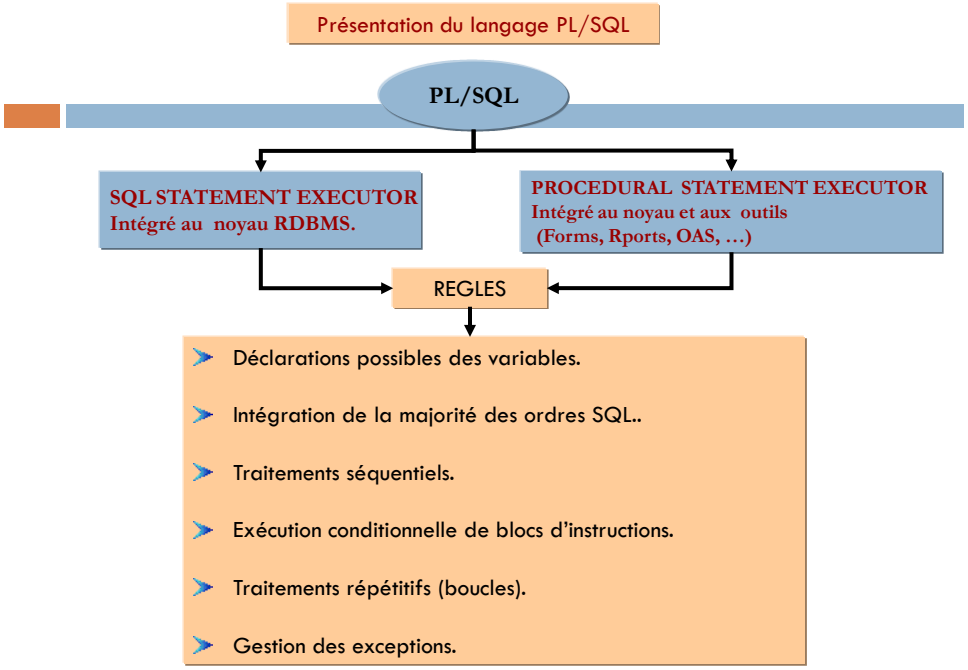
    DROP INDEX emp_ename_idx;

Oracle - PL / SQL

(Procedural Language / Structured Query Language)

Pourquoi PL/SQL ?

- SQL est un langage non procédural
  - Les traitements complexes sont parfois difficiles à écrire si on ne peut utiliser des variables et les structures de programmation comme les boucles et les alternatives
- On ressent vite le besoin d'un langage procédural pour lier plusieurs requêtes SQL avec des variables et dans les structures de programmation habituelles



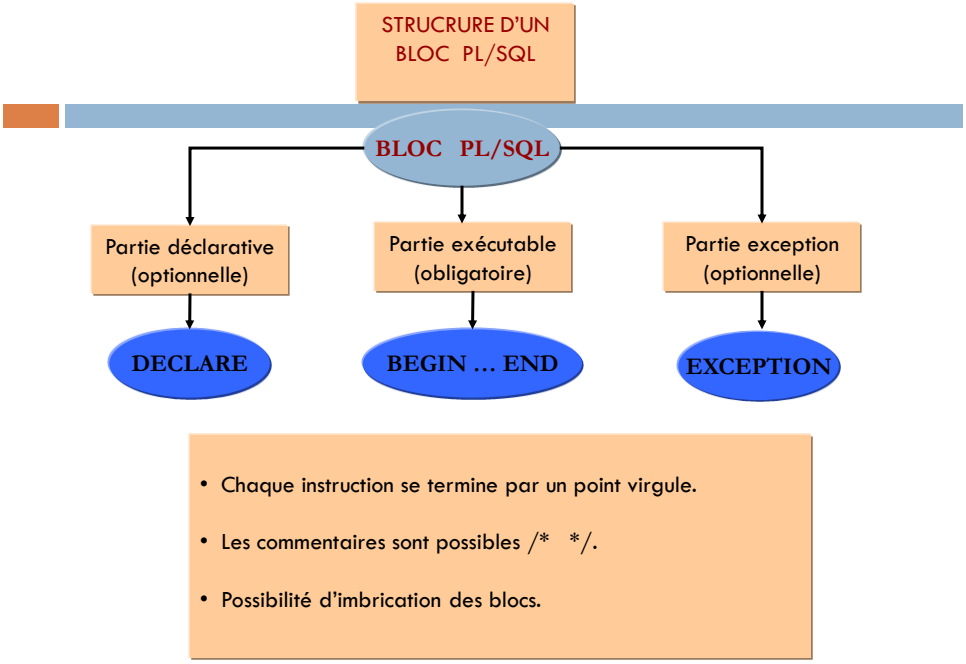
Avantages de PL/SQL

- Intégration
- Amélioration des performances
- Portabilité
- Développement modulaire

CHAPITRE 1: DÉVELOPPER UN BLOC SIMPLE PL/SQL

Structure d'un Block PL/SQL

```
[ DECLARE ]
        - Variables, constantes, curseurs,
          exceptions utilisateurs
BEGIN
        - Ordres SQL
        - Instructions de Contrôle PL/SQL
[ EXCEPTION ]
        - Traitements à effectuer  lors d'erreurs
END ;
```



Règles Syntaxiques d'un Bloc PL/SQL

- Identifiants :
  - Peuvent contenir jusqu'à 30 caractères.
  - Ne peuvent pas contenir de mots réservés à moins qu'ils soient encadrés de guillemets.
  - Doivent commencer par une lettre.
  - Doivent avoir un nom distinct de celui d'une table de la base ou d'une colonne.

Règles Syntaxiques d'un Bloc PL/SQL

- Utiliser un slash (/) pour exécuter un boc PL/SQL anonyme dans PL/SQL.
- Placer un point virgule (;) à la fin d'une instruction SQL ou SQL*PLUS
- Les chaînes de caractères et les dates doivent être entourées de simples quotes (' ').
- Les commentaires peuvent être
  - sur plusieurs lignes avec :
    /* début et
    fin de commentaire*/
  - sur une ligne précédée de :
    -- début et fin de commentaire

Déclaration de Variables et Constantes - Syntaxe

```
identifier [ CONSTANT ] datatype [ NOT NULL ] [ := | DEFAULT expr ];
```

Règles :

- Adopter les conventions de dénomination des objets.
- Initialiser les constantes et les variables déclarées NOT NULL.
- Initialiser les identifiants en utilisant l'opérateur d'affectation (:=) ou le mot réservé DEFAULT.
- Déclarer au plus un identifiant par ligne.

PARTIE DECLARATIVE BLOC PL/SQL

Types classiques

TYPE	VALEURS
BINARY-INTEGER	entiers allant de –2**31 à 2**31)
POSITIVE / NATURAL	entiers positifs allant jusqu'à 2**31 -1
NUMBER	Numérique (entre –2**418 à 2**418)
INTEGER	Entier stocké en binaire (entre –2**126 à 2**126)
CHAR (n)	Chaîne fixe de 1 à 32767 caractères (différent pour une colonne de table)
VARCHAR2 (n)	Chaîne variable (1 à 32767 caractères)
LONG	idem VARCHAR2 (maximum 2 gigaoctets)
DATE	Date (ex. 01/01/1996 ou 01-01-1996 ou 01-JAN-96 ...)
RAW	Permet de stocker des types de données binaire relativement faibles(<= 32767 octets) idem VARCHAR2. Les données RAW ne subissent jamais de conversion de caractères lors de leur transfert entre le programme et la base de données.
LONG RAW	Idem LONG mais avec du binaire

Déclaration de Variables Scalaires - Exemples

```
v_gender        CHAR( 1 );
v_count         BINARY_INTEGER := 0;
v_total_sal             NUMBER( 9, 2 ) := 0;
v_order_date            DATE := SYSDATE;
c_tax_rate              CONSTANT NUMBER ( 3, 2 ) := 8.25;
v_valid         BOOLEAN NOT NULL := TRUE;
```

L'Attribut %TYPE

- Déclarer une variable à partir :
  - D'une autre variable déclarée précédemment
  - De la définition d'une colonne de la base de données
- Préfixer %TYPE avec :
  - La table et la colonne de la base de données
  - Le nom de la variable déclarée précédemment
- PL/SQL détermine le type de donnée et la taille de la variable.

L'Attribut %TYPE - Exemple

```
DECLARE
        v_last_name                 s_emp.last_name%TYPE;
        v_first_name                s_emp.first_name%TYPE;
        v_balance                   NUMBER( 7, 2 );
        v_minimum_balancev_balance%TYPE := 10;
```

- Le type de données de la colonne peut être inconnu.
- Le type de données de la colonne peut changer en exécution.

L'Attribut %ROWTYPE - Avantages

- Le nombre de colonnes, ainsi que les types de données des colonnes de la table de référence peuvent être inconnus.
- Le nombre de colonnes, ainsi que le type des colonnes de la table de référence peuvent changer en exécution
- Utile lorsqu'on recherche
  - Une ligne avec l'ordre SELECT.
  - Plusieurs lignes avec un curseur explicite.

Exemple

```
DECLARE
        dept_record     s_dept%ROWTYPE;
        emp_record      s_emp%ROWTYPE;
```

Les variables référencées à une table de la base

- Elles sont liées à des tables au niveau de la base.
- On les déclare par l'attribut : %ROWTYPE

Exemples

```
DECLARE
    agent  employe%ROWTYPE          -- employe est la table employe
--- de la base.
Au niveau traitement, on pourra écrire :
BEGIN
        SELECT *                    -- Sélection de tous les -- champs
        INTO  agent
        FROM  employe
        WHERE nom='DUMAS';
END;
```

Ou

```
BEGIN
        SELECT    nom,dt_entree     -- Sélection de certains champs
        INTO  agent.nom, agent.dt_entree
        FROM  employe
        WHERE nom='DUMAS';
END;
```

Opérateurs en PL/SQL

- Logiques
- Arithmétiques
- Concaténation

} Identiques à SQL

- Opérateur exponentiel (**)
- Parenthèses pour contrôler l'ordre des opérations

Fonctions en PL/SQL - Exemples

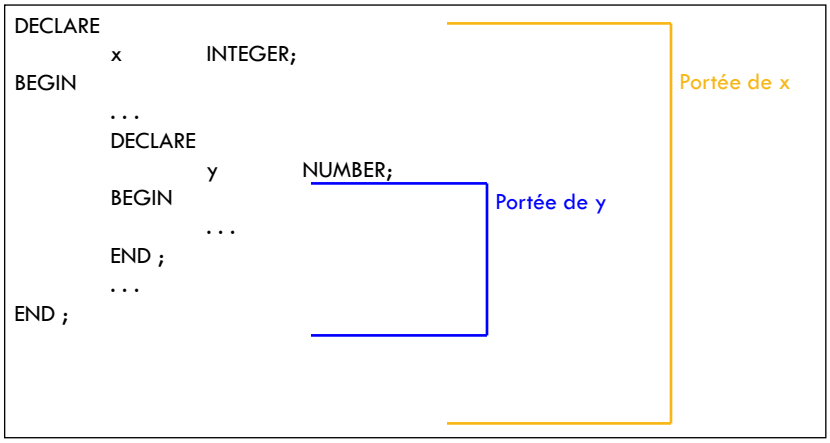
- Construire une liste d'adresses pour une société :

```
v_mailing_address := v_name || CHR( 10 ) ||
          v_address || CHR( 10 ) || v_country || CHR ( 10 ) ||
          v_zip_code
```

- Convertir le nom de famille en majuscule :

```
v_last_name          := UPPER( v_last_name ) ;
```

Blocs Imbriqués et Portée d'une Variable - Exemple



Conventions de Casse pour le Code

Conventions	Exemples et Casse
Commandes SQL	SELECT, INSERT
Mots Clés PL/SQL	DECLARE, BEGIN, IF
Types de Données	VARCHAR2, BOOLEAN
Identifiants et Paramètres	v_sal, emp_cursor, g_sal
Tables et Colonnes de la base de données	s_emp, order_date, id

CHAPITRE 2: INTERACTION AVEC ORACLE

Retrouver des Données (Extraire de données) - Syntaxe

Retrouver des lignes de la base de données avec le SELECT

```
SELECT    select_list
INTO              variable_name | record_name
FROM              table
WHERE     condition ;
```

- La clause INTO est obligatoire.
- Une seule ligne doit être retournée.
- Toute la syntaxe du SELECT est disponible.

Retrouver des Données - Exemple

Retrouver toutes les informations d'un département donné.

```
DECLARE
    v_nom emp.nome%TYPE;
    v_emp emp%ROWTYPE;
BEGIN
    select nome into v_nom
    from emp
    where matr = 500;

     select * into v_emp
     from emp
     where matr = 500;
END
/
```

Exceptions SELECT

- Les ordres SELECT en PL/SQL doivent ramener une et une seule ligne.
- Si aucune ou plusieurs lignes sont retrouvées une exception est déclenchée.
- Exceptions du SELECT :
  - **TOO_MANY_ROWS**
  - **NO_DATA_FOUND**

- Les requêtes SQL (insert, update, delete,…) peuvent utiliser les variables PL/SQL
- Les commit et rollback doivent être explicites ; aucun n'est effectué automatiquement à la sortie d'un bloc
- Voyons plus de détails pour l'insertion de données

Insertion de Données - Exemple

Ajouter des nouveaux employées à la base de donné :

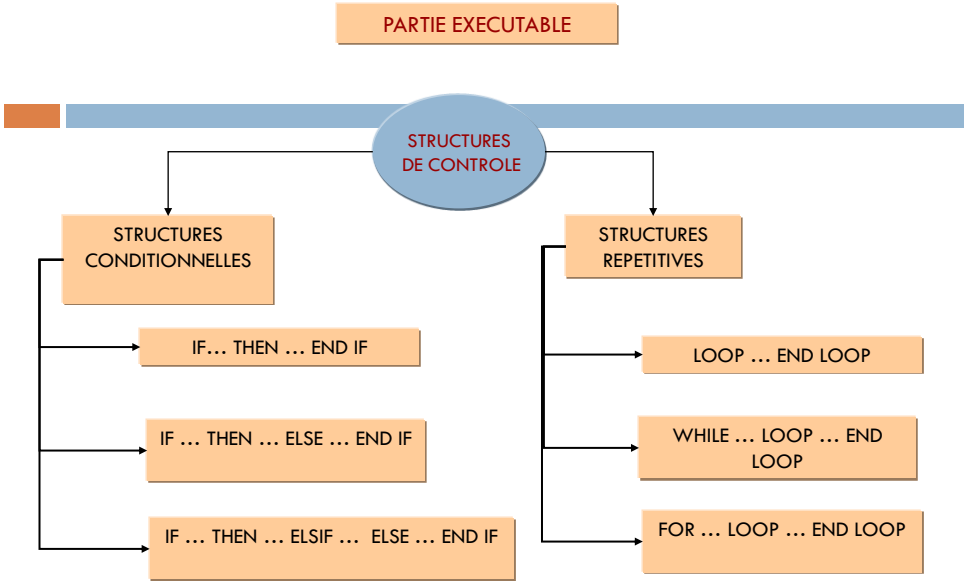
```
DECLARE
      v_emp emp%ROWTYPE;
      v_nom emp.nome%TYPE;
BEGIN
      v_nom := 'ALAMI';
      insert into emp (matr, nome)
      values(600, v_nom);

      v_emp.matr := 610;
      v_emp.nome := 'TOTO';
      insert into emp (matr, nome)
      values(v_emp.matr, v_emp.nome);
      commit;
END; --Fin du bloc PL --
/
```

CHAPITRE 3: TRAITEMENTS CONDITIONNELS ET TRAITEMENTS RÉPÉTITIFS

Contrôler le Déroulement d'une Exécution PL/SQL

Modifier le déroulement logique des instructions en utilisant des structures de contrôle

- Structures de contrôle conditionnel (Instruction IF)
- Structures de Contrôle Itératif
  - Boucle de base
  - Boucle FOR
  - Boucle WHILE
  - Instruction EXIT



L'Instruction IF - Syntaxe

- On peut déclencher des actions en fonction du résultat de conditions

```
IF condition THEN
        instructions ;
[ ELSIF conditions THEN
        instructions ; ]
[ ELSE
        instructions ;]
END IF;
```

- ELSIF en un mot
- END IF en deux mots
- une seule clause ELSE est permise

Exemple

```
IF salaire < =1000 THEN
   nouveau_salaire := ancien_salaire + 100;
ELSEIF salaire > 1000 AND emp_id_emp = 1 THEN
   nouveau_salaire := ancien_salaire + 500;
ELSE   nouveau_salaire := ancien_salaire + 300;
END IF;
```

Choix

```
CASE expression
WHEN expr1 THEN instructions1;
WHEN expr2 THEN instructions2;
…
ELSE instructionsN;
END CASE;
```

Expression de type simple

Instructions LOOP

- Les boucles répètent une instruction ou un ensemble d'instructions plusieurs fois.
- Trois types de boucles
  - Boucle de base
  - Boucle FOR
  - Boucle WHILE
- L'instruction EXIT permet de sortir de la boucle

Boucle de Base

```
LOOP                                          -- Début de boucle
        instruction1 ;                        -- Instructions
        . . .
        EXIT [ WHEN condition ];      -- Sortie de boucle
END LOOP ;                                    -- Fin de boucle
```

Exercice

Ecrire un programme PL/SQL qui affiche les multiples de 3, 4 et 5 qui sont entre 4 et 32.

```
SET SERVEROUTPUT  ON  -- sous SQL pLUS
DECLARE
        i NUMBER(2) := 4;
BEGIN
        LOOP
                IF (MOD(i,3)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 3');
                END IF;
                IF (MOD(i,4)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 4');
                END IF;
                IF (MOD(i,5)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 5');
                END IF;
                i := i+1;
                EXIT WHEN i>32;
        END LOOP;
END;
/
```

Boucle FOR - Exemple

- Afficher le nombre de fois où la boucle est exécutée et la dernière valeur de l'index.

```
for i IN 1..100 LOOP
somme := somme + i;
end loop
/
```

Boucle WHILE - Exemple

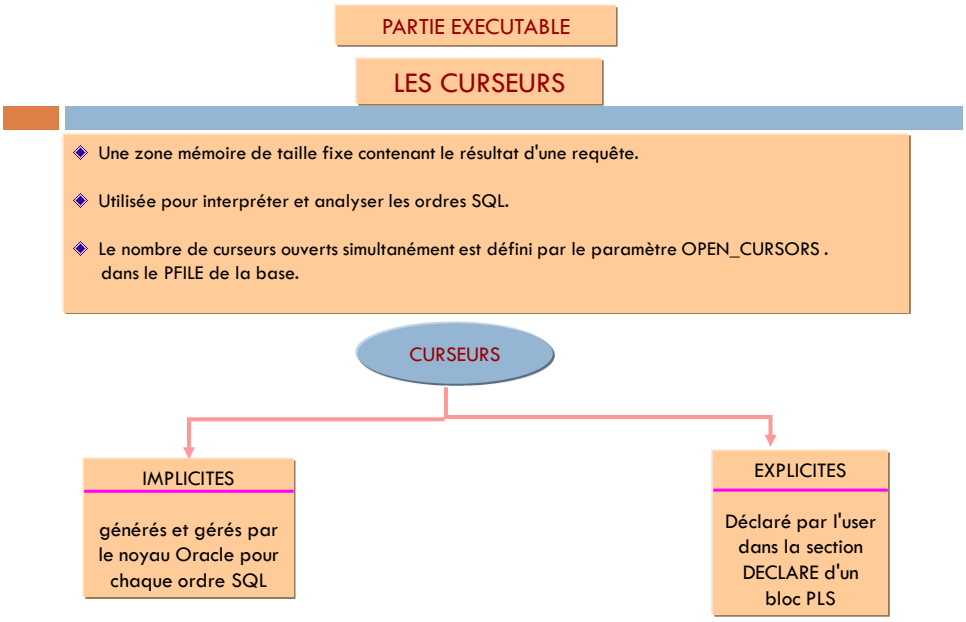
```
SET SERVEROUTPUT  ON  -- sous SQL pLUS
DECLARE
        i NUMBER(2) := 4;
BEGIN
        WHILE i<33   LOOP
                IF (MOD(i,3)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 3');
                END IF;
                IF (MOD(i,4)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 4');
                END IF;
                IF (MOD(i,5)=0) THEN
DBMS_OUTPUT.PUT_LINE (i || ' est un multiple de 5');
                END IF;
                i := i+1;
        END LOOP;
END;
/
```

CHAPITRE 4: CURSEURS

Qu'est ce qu'un Curseur ?

- Le serveur Oracle utilise des zone de travail appelées *Zone Sql Privées* pour exécuter les instructions SQL et pour stocker les informations en cours de traitement.

- Vous pouvez utilisez des curseurs PL/SQL pour nommer une zone SQL privée et accéder aux données qu'elle contient.



Curseurs

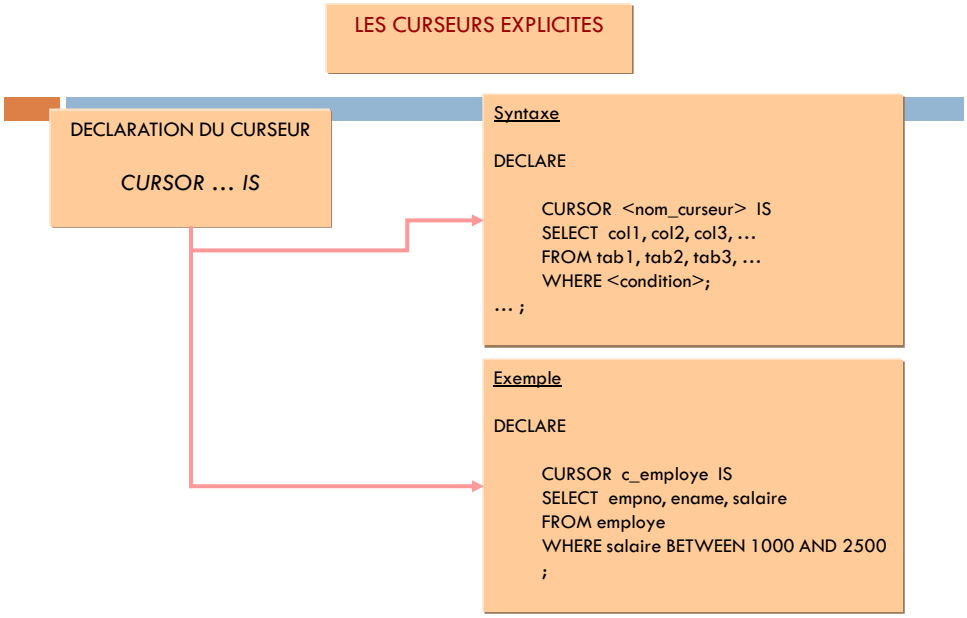
L'utilisation d'un curseur nécessite 4 étapes :

1. Déclaration du curseur : Section DECLARE
2. Ouverture du curseur : Section BEGIN
3. Traitement des lignes : Section BEGIN
4. Fermeture du curseur : Section BEGIN OU EXCEPTION

La déclaration d'un curseur

- La déclaration du curseur permet de stocker l'ordre Select dans le curseur.
- Le curseur se définit dans la partie Declare d'un bloc PL/Sql.

Cursor nomcurseur IS Requete_SELECT ;



La déclaration d'un curseur

Declare

Cursor DEPT10 is

select ename, sal from emp where deptno=10 order by sal ;

Begin

… …;

End ;

L'ouverture du curseur

- L'ouverture du curseur réalise :
  1. l'allocation mémoire du curseur
  2. l'analyse sémantique et syntaxique de l'ordre
  3. le positionnement de verrous éventuels (si select for update...)
- C'est seulement à l'ouverture du curseur que la requête SQL s'éxécute.
- L'ouverture du curseur se fait dans la section Begin du Bloc.

  OPEN nomcurseur ;

L'ouverture du curseur

Declare

Cursor DEPT10 is

select ename, sal from emp where deptno=10 order by sal ;

Begin

...Open DEPT10;

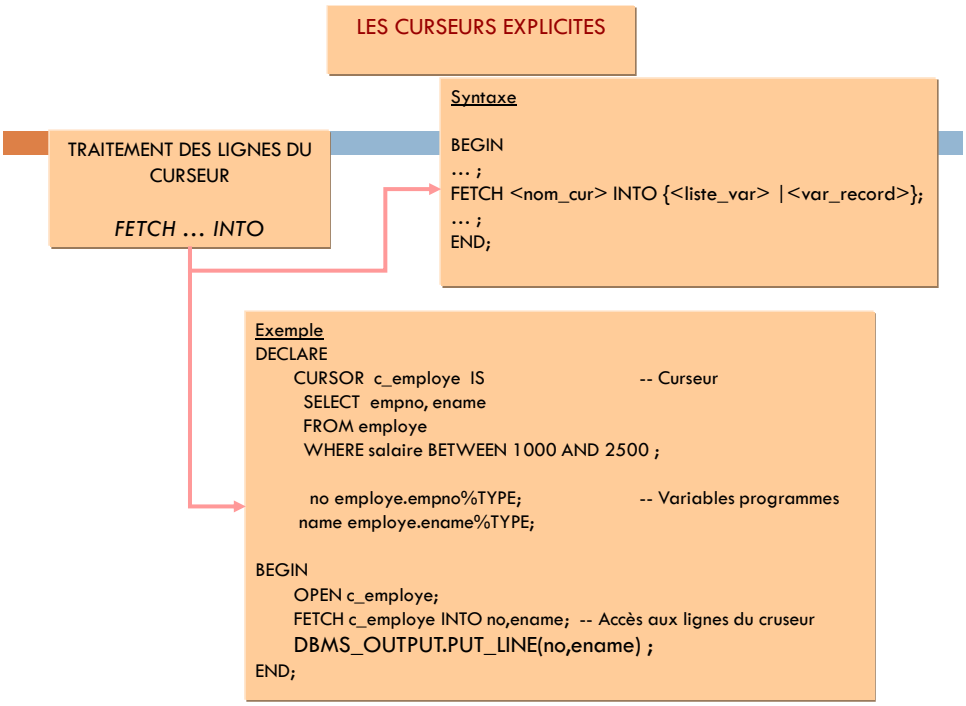
.....

End ;

Traitement des lignes

- Après l'exécution du Select
  - les lignes ramenées sont traitées une par une,
  - la valeur de chaque colonne du Select doit être stockée dans une variable réceptrice définie dans la partie Declare du bloc.
  - Le fetch ramène une seule ligne à la fois,
  - pour traiter n lignes il faut une boucle.

FETCH nomcurseur INTO liste_variables ou Nom_enregistrement;



```
create table resultat (nom1 char(10), sal1 number(7,2))
/
Declare
Cursor DEPT10 is select ename, sal from emp where deptno=20 order by sal ;
-- variables réceptrices
nom emp.ename%TYPE; -- Variable locale de même type que le champ ename
salaire emp.sal%TYPE;
Begin
Open DEPT10;
Fetch DEPT10 into nom, salaire ; -- Lecture 1° tuple
WHILE DEPT10%found loop -- Tant qu'on trouve une ligne
If salaire > 2500 then
insert into resultat values (nom,salaire);
end if;
Fetch DEPT10 into nom,salaire ; -- Lecture tuple suivant
end loop;
Close DEPT10;
End ;
/
```

Attributs d'un Curseur Explicite

Obtenir des informations sur le curseur en utilisant les attributs de curseur.

Attribut	Type	Description
%ISOPEN	Booléen	Evalué à TRUE si le curseur est ouvert.
%NOTFOUND	Booléen	Evalué à TRUE si le dernier fetch n'a ramené aucune ligne
%FOUND	Booléen	Evalué à TRUE tant que le fetch ramène des lignes
%ROWCOUNT	Numérique	Evalué au nombre total de lignes ramenées jusqu'à présent.

Attributs d'un Curseur Explicite

- Les attributs d'un curseur sont des indicateurs sur l'état d'un curseur. Ils nous fournissent des informations quant à l'exécution de l'ordre.
- Elles sont conservées par PI/Sql après l'exécution du curseur.

La fermeture du curseur

- Après le traitement des lignes, l'étape de fermeture permet d'effectuer la libération de la place mémoire.

CLOSE nomcurseur; ***Close dept10 ;***

```
create table resultat (nom1 char(35), sal1 number(8,2))
/
Declare
Cursor C1 is select * from pilote where adresse='Paris';
-- variable réceptrice
unpilot pilote%rowtype;
Begin
Open C1;
Fetch c1 into unpilot ; -- Lecture 1° tuple
WHILE C1%found
loop
If unpilot.comm is not null then
insert into resultat values (unpilot.nompilot, unpilot.salpilot);
end if;
Fetch c1 into unpilot ; -- Lecture tuple suivant
end loop;
Close c1;
End ;
/
```

Boucle LOOP pour un curseur

```
BEGIN
open salaires;
loop
    fetch salaires into salaire;
    exit when salaires%notfound;
    if salaire is not null then
            total := total + salaire;
            DBMS_OUTPUT.put_line(total);
    end if;
end loop;
close salaires; -- Ne pas oublier
DBMS_OUTPUT.put_line(total);
END;
```

Boucle FOR pour un curseur

```
declare
cursor c is
select dept, nome from emp
where dept = 10;
employe c%rowtype;
begin
FOR employe IN c LOOP
dbms_output.put_line(employe.nome);
END LOOP;
end;
```

MODIFICATION DE DONNEES

- Les modifications de données s'effectuent normalement par les instructions SQL : INSERT, UPDATE et DELETE

- PL/Sql permet la possibilité d'utiliser l'option

  CURRENT OF nom_curseur dans la clause WHERE des instructions UPDATE et DELETE. Cette option permet de modifier ou de supprimer la ligne distribuée par la commande FETCH.

  - Pour utiliser cette option, il faut ajouter la clause FOR UPDATE à la fin de la définition du curseur.

```
DECLARE
Cursor C1 is
select ename, sal from emp
for update of sal;
resC1 c1%rowtype;
BEGIN
Open C1;
Fetch C1 into resC1;
While C1%found Loop
      If resC1.sal > 1500 then
           update emp
           set sal = sal * 1.1
          where current of c1;
      end if;
   Fetch C1 into resC1;
end loop;
close C1 ;
END;
/
```

FOR UPDATE

- **FOR UPDATE [OF *col1, col2,…]***
- Cette clause bloque toute la ligne ou seulement les colonnes spécifiées Les autres transactions ne pourront modifier les valeurs tant que le curseur n'aura pas quitté cette ligne

Clause WHERE CURRENT OF - Exemple

Modifier les lignes correspondantes à un critère avec un curseur.

```
. . .
        CURSOR emp_cursor IS
                SELECT ...
                FOR UPDATE;
BEGIN
        . . .
        FOR emp_record IN emp_cursor LOOP
                UPDATE . . .
                WHERE CURRENT OF emp_cursor ;
        . . .
        END LOOP ;
        COMMIT ;
END ;
```

GESTION DES ERREURS (EXCEPTION)

Exception

- Une exception est une erreur qui survient durant une exécution
- 2 types d'exception :
  - **Interne**
    - **exception oracle pré-définie**
    - **exception oracle non pré-définie**
  - **Externe (exception définie par l'utilisateur)**

Exception

- **Les exceptions internes** sont générées par le moteur du système (division par zéro, connexion non établie, table inexistante, privilèges insuffisants, mémoire saturée, espace disque insuffisant, ...).
  - Une erreur interne est produite quand un bloc PL/Sql viole une règle d'Oracle ou dépasse une limite dépendant du système d'exploitation.
  - Chaque erreur ORACLE correspond un code SQL (SQLCODE)
- **Les exceptions externes** sont générées par l'utilisateur (stock à zéro, ...).

Exception

EXCEPTION	DESCRIPTION	TRAITEMENT
Erreur pré-définie du oracle	Une des 20 erreurs qui arrivent le plus fréquemment en langage P/SQL	Ne pas la déclarer et laisser oracle serveur l'émettre implicitement
Erreur non pré-définie du oracle	Toutes les autres erreur standard d'oracle	La déclarer à l'intérieur de la section declare et laisser oracle l'émettre implicitement
Erreur définie par l'utilisateur	Une condition que le programmeur définit comme anomalie	La déclarer à l'intérieur de la section declare et son émission est explicite

Intercepter les Exceptions - Syntaxe

```
EXCEPTION
  WHEN exception1 [OR exception2 . . .] THEN
    statement1;
    statement2;
    . . .
  [WHEN exception3 [OR exception4 . . .] THEN
    statement1;
    statement2;
    . . .]
  [WHEN OTHERS THEN
    statement1;
    statement2;
    . . .]
```

Règles pour intercepter les Exceptions

- Le mot clé EXCEPTION débute la section de la gestion des exceptions
- Plusieurs exceptions sont permises (définie et pré-définie)
- Une seule exception est exécutée avant de sortir d'un bloc
- WHEN OTHERS est la dernière clause
  - Intercepte toutes les exceptions non gérées dans la même section d'exception
  - Utilisez le gestionnaire d'erreurs OTHERS et placez le en dernier lieu après tous les autres gestionnaire d'erreurs, sinon il interceptera toutes les exceptions mêmes celle qui sont prédéfinies.

Les erreurs internes d'Oracle pré-définies

- Utiliser le nom standard à l'intérieur de la section Exception
- Les noms des exceptions pré-définies oracle sont regroupées dans ce tableau :

Les erreurs internes d'Oracle pré-définies

Nom d'exception	Erreur ORACLE	SQLCODE
CURSOR_ALREADY_OPEN	ORA-06511	-6511
DUP_VAL_ON_INDEX	ORA-00001	-1
INVALID_CURSOR	ORA-01001	-1001
INVALID_NUMBER	ORA-01722	-1722
LOGIN_DENIED	ORA-01017	-1017
NO_DATA_FOUND	ORA-01403	+100
NOT_LOGGED_ON	ORA-01012	-1012
PROGRAM_ERROR	ORA-06501	-6501
ROWTYPE_MISMATCH	ORA-06504	-6504
STORAGE_ERROR	ORA-06500	-6500
TIMEOUT_ON_RESOURCE	ORA-00051	-51
TOO_MANY_ROWS	ORA-01422	-1422
VALUE_ERROR	ORA-06502	-6502
ZERO_DIVIDE	ORA-01476	-1476

Les erreurs internes d'Oracle pré-définies

- CURSOR_ALREADY_OPEN : tentative d'ouverture d'un curseur déjà ouvert..
- DUP_VAL_ON_INDEX: insertion d'une ligne en doublon
- INVALID_CURSOR : opération incorrecte sur un curseur, comme par exemple la fermeture d'un curseur qui n'a pas été ouvert.
- LOGIN_DENIED : connexion à la base échouée car le nom utilisateur ou le mot de passe est invalide.

Les erreurs internes d'Oracle pré-définies

- NO_DATA_FOUND : déclenché si la commande SELECT INTO ne retourne aucune ligne ou si on fait référence à un enregistrement non initialise d'un tableau PL/SQL.
- NOT_LOGGED_ON : tentative d'accès à la base sans être connecté.
- PROGRAM_ERROR : problème général dû au PL/SQL.

Les erreurs internes d'Oracle pré-définies

- ROWTYPE_MISMATCH : survient lorsque une variable curseur d'un programme hôte retourne une valeur dans une variable curseur d'un bloc PL/SQL qui n'a pas le même type.
- STORAGE_ERROR : problème de ressources mémoire dû à PL/SQL.
- TIMEOUT_ON_RESOURCE : dépassement du temps dans l'attente de libération des ressources (lié aux paramètres de la base).

Les erreurs internes d'Oracle pré-définies

- TOO_MANY_ROWS : la commande SELECT INTO retourne plus d'une ligne.
- VALUE_ERROR : erreur arithmétique, de conversion, ou de contrainte de taille.
- ZERO_DIVIDE : tentative de division par zéro.

Exemple

```
DECLARE
    v_sal emp.sal%type;
BEGIN
    SELECT sal  INTO v_sal  from emp;
EXCEPTION
    WHEN TOO_MANY_ROWS then ... ;
    -- gérer erreur trop de lignes
    WHEN NO_DATA_FOUND then ... ;
    -- gérer erreur pas de ligne
    WHEN OTHERS then ... ;
    -- gérer toutes les autres erreurs
END ;
```

Exceptions Oracle Non Prédéfinies

- Vous pouvez intercepter une erreur oracle non pré-définie en la déclarant en préalable, ou en utilisant la commande OTHERS
- L'exception déclarée et implicitement déclenchée

Exceptions Utilisateur **(externes)**

- PL/SQL permet à l'utilisateur de définir ses propres exceptions.
- La gestion des anomalies utilisateur peut se faire dans un bloc PL/SQL en effectuant les opérations suivantes :

Exceptions Utilisateur

1. Nommer l'erreur (type exception) dans la partie Declare du bloc.

   DECLARE

   Nom_ano Exception;

2. Déterminer l'erreur et passer la main au traitement approprié par la commande **Raise.**

   BEGIN

   If (condition_anomalie) then raise Nom_ano ;

3. Effectuer le traitement défini dans la partie EXCEPTION du Bloc.

   EXCEPTION

   WHEN (Nom_ano) then (traitement);

Exceptions Utilisateur

```
DECLARE
   ...
   Nom_ano EXCEPTION;
BEGIN
   instructions ;
   IF (condition_anomalie) THEN RAISE Nom_ano;
   ...
EXCEPTION
   WHEN Nom_ano THEN (traitement);
END ;
```

On sort du bloc après l'exécution du traitement d'erreur.

```
DECLARE
Erreur_comm exception ;
v_pilot pilote%rowtype ;
BEGIN
Select * into v_pilot From Pilote
Where nopilot = '7100' ;
If  v_pilot.comm > v.pilot.sal Then
Raise erreur_comm ;
.....
EXECPTION
When erreur_comm then
Insert into erreur values(v_pilot.nom, ' Commission > salaire') ;
When NO_DATA_FOUND Then
Insert into erreur values(v_pilot.nopilot, ' non trouvé') ;
END ;
```

Fonctions d'interception des erreurs

- SQLCODE

  Renvoie la valeur numérique associé au code de l'erreur.

  Vous pouverz l'assigneer à une variable de type number

- SQLERRM

  Renvoie le message associé au code de l'erreur.

Exemple de SQLCODE

Valeur de SQLCODE	Description
0	Pas d'exception enregistrée
1	Exception définie par l'utilisateur
+100	Exception NO_DATA_FOUND
Négative number	Autre code d'erreur oracle

Fonctions d'interception des erreurs- Exemple

Lorsqu'une exception est intercetée par la clause WHEN OTHERS, vous pouvez utiliser un ensemble de fonctions standard pour identifier l'erreur.

```
DECLARE
  v_error_code      NUMBER;
  v_error_message   VARCHAR2(255);
BEGIN
. . .
EXCEPTION
. . .
  WHEN OTHERS THEN
    ROLLBACK;
    v_error_code := SQLCODE;
    v_error_message := SQLERRM;
    insert into erreur values(v_error_code, v_error_message);
END;
```

PROCEDURES, FONCTIONS ET PACKAGES

Généralité

- Une **procédure est un bloc PL/SQL nommé.**
- Une **fonction est une procédure qui retourne une valeur.**
- Un **package est un agrégat de procédures et de fonctions.**

Généralité

- Les packages, procédures, ou fonctions peuvent être appelés depuis toutes les applications qui possèdent une interface avec ORACLE (SQL*PLUS, Pro*C, SQL*Forms, ou un outil client particulier comme NSDK par exemple).

Généralité

- Les procédures (fonctions) permettent de :
  - Réduire le trafic sur le réseau (les procédures sont locales sur le serveur)
  - Mettre en œuvre une architecture client/serveur de procédures et rendre indépendant le code client de celui des procédures (à l'API près)
  - Masquer la complexité du code SQL (simple appel de procédure avec passage d'arguments)
  - Optimiser le code (les procédures sont compilées avant l'exécution du programme et elles sont exécutées immédiatement si elles se trouvent dans la SGA (zone mémoire gérée par ORACLE). De plus une procédure peut être exécutée par plusieurs utilisateurs.

Généralité

- Les packages permettent :
  - de regrouper des procédures ou des fonctions (ou les deux). On évite ainsi d'avoir autant de sources que de procédures.
  - De travailler en équipes et l'architecture applicative peuvent donc plus facilement s'organiser du côté serveur, où les packages regrouperont des procédures à forte cohésion intra (Sélection de tous les articles, Sélection d'un article, Mise à jour d'un article, Suppression d'un article, Ajout d'un article).
- Les packages sont utilisés comme de simples librairies par les programmes clients (librairies distantes « sur le serveur »)

Généralité

- Les équipes de développement doivent prendre garde à ne pas travailler chacune dans « leur coin ».
- Les développeurs ne doivent pas perdre de vue la logique globale de l'application et les scénarios d'activité des opérateurs de saisie.
  - A l'extrême, on peut finir par coder une procédure extrêmement sophistiquée qui n'est sollicitée qu'une fois par an pendant une seconde. Ou encore, une gestion complexe de verrous pour des accès concurrent qui n'ont quasiment jamais lieu.

Procédures

- Les procédures ont un ensemble de paramètres modifiables en entrée et en sortie.

```
CREATE [ OR REPLACE ] PROCEDURE [<user>].<nom_proc>
 (arg1 IN type1 [DEFAULT val_initiale [, arg2 OUT type2 [, arg3 IN OUT type3, …])
 AS

    [ Déclarations des variables locales ]
EXCEPTION
BEGIN
```

Procédures

Argument	Signification
IN	Valeur par défaut. Argument en entrée. Elle ne être modifiée par la procédure.
OUT	Argument en sortie (modifié par la prcédure).
IN OUT	Argument en entrée sortie. Elle peut être lue et modifiée par la procédure.
TYPE	Type de l'argument sans spécification de la taille.
DEFAULT	Affecte une valeur par défaut à l'argument.
IS	Permet la définition de la procédure.

Procédure Exemple

Compter le nombre d'employés pour un département donné.

```
CREATE  OR  REPLACE  PROCEDURE   proc_dept (p_no IN dept.deptno%TYPE)
IS
     v_no NUMBER;
 BEGIN
     SELECT  COUNT(deptno)
     INTO v_no
     FROM  emp
     WHERE deptno=p_no;
     DBMS_OUTPUT.PUT_LINE('Nombre d'employés : '||' '||v_no);
 END;
 /
```

Procédure Exemple

- Exemple de procédure qui modifie le salaire d'un employé.
  - Arguments : Identifiant de l'employée, Taux

Procédure Exemple

modifie_salaire.sql

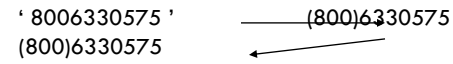
```
create procedure modifie_salaire
(id in number, taux in number) is
begin
update employe set salaire=salaire*(1+taux)
where Id_emp= id;
exception
when no_data_found then
DBMS_OUTPUT.PUT_LINE('Employé inconnu : ' ||to_char(id));
End;
/
```

PROCEDURES

```
SQL>CREATE OR REPLACE PROCEDURE query_emp
2  (v_id IN emp.empno%TYPE,
2  v_name  OUT  emp.ename%TYPE,
3  v_salary  OUT  emp.sal%TYPE,
4  v_comm   OUT  emp.comm%TYPE)
5  IS
6  BEGIN
7          SELECT ename,sal,comm
8          INTO v_name,v_salary, v_comm
9          FROM emp
10         WHERE empno = v_id;
11 END query_emp;
12 /
Procedure created.
```

PROCEDURES / PARAMETRES

- EXEMPLE IN OUT




```
SQL> CREATE OR REPLACE PROCEDURE format_phone
2 (v_phone_no IN OUT VARCHAR2(12))
3  IS
4  BEGIN
5    v_phone_no := ‘ (‘ ||substr(v_phone_no,1,3)|| ’) ’||substr(v_phone,4,7)
6  END format_phone;
7  /
```

Compilation de la procédure modifie_salaire

- Il faut compiler le fichier sql qui s’appelle ici modifie_salaire.sql (attention dans cet exemple le nom du script correspond à celui de la procédure, c’est bien le nom du script sql qu’il faut passer en argument à la commande start).

SQL> **start modifie_salaire**

Procedure created.

Appel de la procédure modifie_salaire

```
SQL> begin
2 modifie_salaire (15,-0.5);
3 end;
4 /
```

Appel de la procédure modifie_salaire

SQL> begin
2 modifie_salaire (15,-0.5);
3 end;
4 /

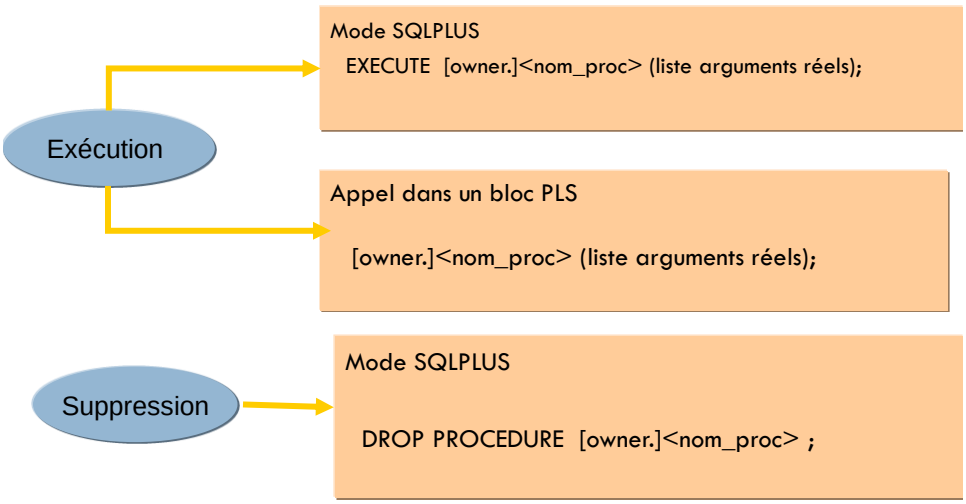
Appel de la procédure modifie_salaire

- L'utilisation d'un script qui contient les 4 lignes précédentes est bien sûr également possible : demarre.sql

```
begin
modifie_salaire (15,-0.5);
end;
/
```

- Lancement du script demarre.sql :

```
SQL> start demarre
```



Correction des erreurs

- Si le script contient des erreurs, la commande ***show err permet de visualiser les erreurs.***
- Pour visualiser le script global :
  - commande l (lettre l)
  - pour visualiser la ligne 4 : commande l4

Correction des erreurs

```
SQL> start modifie_salaire

Warning: Procedure created with compilation errors.

SQL> show err
Errors for PROCEDURE MODIFIE_SALAIRE:

LINE/COL ERROR
-------- -----------------------------------------------------------
4/8      PLS-00103: Encountered the symbol "VOYAGE" when expecting one of the
following:
         := . ( @ % ;
         Resuming parse at line 5, column 21.

SQL> l4

  4*    update employe set salaire=salaire*(1+taux)
```

Fonctions

- Une fonction est une procédure qui retourne une valeur. La seule différence syntaxique par rapport à une procédure se traduit par la présence du mot clé RETURN.
- Une fonction précise le type de donnée qu'elle retourne dans son prototype (signature de la fonction).
- Le retour d'une valeur se traduit par l'instruction RETURN (valeur).

Fonctions

```
CREATE  [ OR  REPLACE ] FUNCTION  [<user>].<nom_proc>
 (arg1 IN type1 [DEFAULT val_initiale [, arg2 IN type2, …])
 RETURN  type_retour AS

      [ Déclarations des variables locales ]

BEGIN
      Contenu du bloc PLS
      RETURN  (var_retour );
Exception
END  [<nom_proc>];
/
```

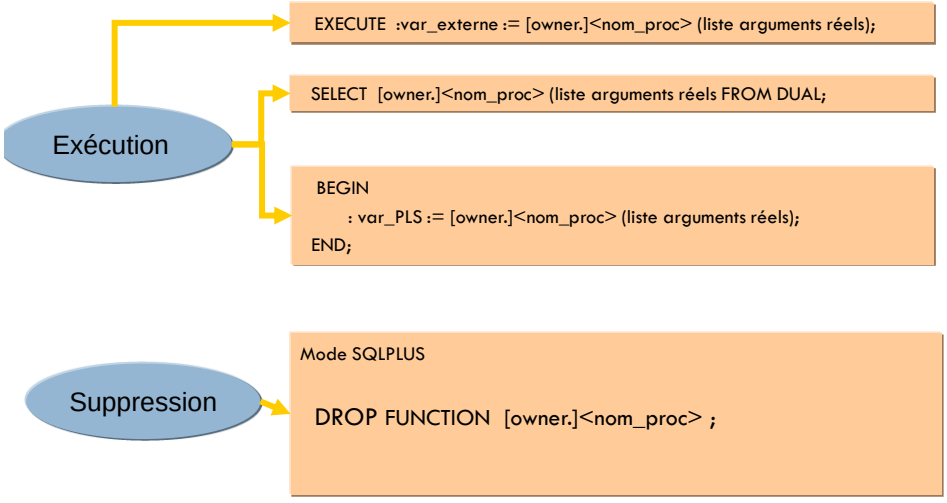
Fonction -Exemple

- Compter le nombre d'employés pour un département donné.

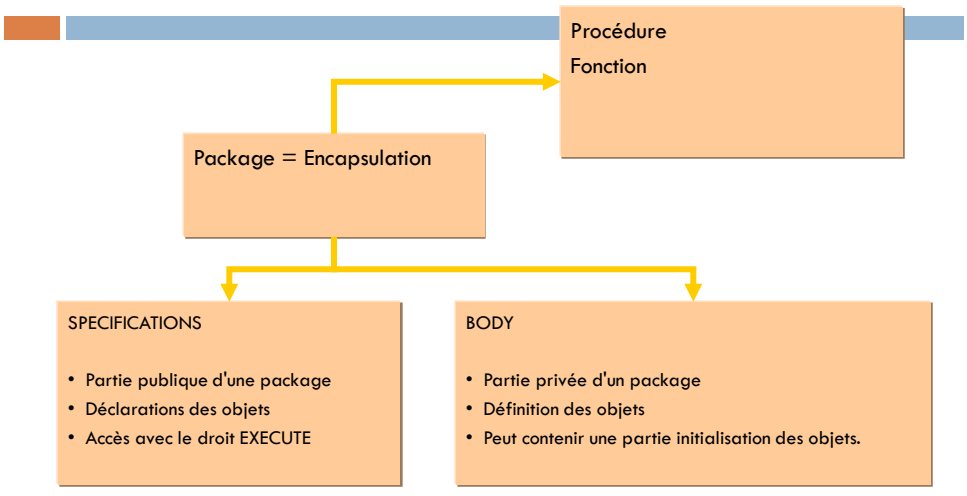
Fonction -Exemple

```
CREATE  OR  REPLACE  FUNCTION   proc_dept (p_no IN dept.deptno%TYPE)
RETURN NUMBER AS
     v_no NUMBER;
 BEGIN
     SELECT  COUNT(deptno)
     INTO v_no
     FROM  emp
     WHERE deptno=p_no;
     RETRUN (v_no);
 END;
 /
```

LES FONCTIONS - Opérations de base



LES PACKAGES



La structure Générale d'un package

package_general.sql

CREATE OR REPLACE PACKAGE **nom_package IS**
définitions des types utilisés dans le package;
prototypes de toutes les procédures et fonctions du package;
END **nom_package;**
/
CREATE OR REPLACE PACKAGE BODY nom_package IS
déclaration de variables globales;
définition de la première procédure;
définition de la deuxième procédure;
etc. ...
END nom_package;
/

La structure Générale d'un package

- Un package est composé d'un en tête et d'un corps :
  - L'en tête comporte les types de données définis et les prototypes de toutes les procédures (fonctions) du package.
  - Le corps correspond à l'implémentation (définition) des procédures (fonctions).

La structure Générale d'un package

- Le premier END marque la fin de l'en tête du package. Cette partie doit se terminer par / pour que l'en tête soit compilé.
- Le corps (body) du package consiste à implémenter (définir) l'ensemble des procédures ou fonctions qui le constitue. Chaque procédure est définie normalement avec ses clauses BEGIN ... END.
- Le package se termine par / sur la dernière ligne.

exemple

- Package *paquet1* comportant deux procédures (augmentation de salaire et suppression de vendeur) :

```
create or replace package ges emp is

procedure augmente salaire (v Id emp in number,
                            v taux salaire in number);

function moyenne_salaire (v_Id_employe IN NUMBER)
return NUMBER;

end ges emp;
/

create or replace package body ges_emp is

procedure augmente salaire (v Id emp in number,
                            v taux salaire in number)
is

begin

      update employe set salaire= salaire * v taux salaire
      where Id emp= v Id emp;
      commit;

end augmente_salaire;


function moyenne salaire (v Id employe IN NUMBER)
return NUMBER
is
valeur NUMBER;

begin

      select avg(salaire)
      into valeur
      from employe
      groupe by emp id emp;
      return (valeur);
end moyenne_salaire;


end ges_emp;
/
```

Compilation du package paquet1

SQL> @paquet1

Package created.

Package body created

Opérations sur les packages

- **Exécution de la procédure augmente_salaire du package ges_emp**

```
SQL> begin
        2 ges_emp.augmente_salaire(4,50);
        3 end;
        4 /
PL/SQL procedure successfully completed.
```

Opérations sur les packages

- **Exécution de la procédure augmente_salaire du package ges_emp**

```
SQL> begin
        2 ges_emp.augmente_salaire(4,50);
        3 end;
        4 /
PL/SQL procedure successfully completed.
```

TRIGGERs

Déclencheurs : définition

- Les triggers sont des simples procédures stockées qui s'exécutent implicitement lorsqu'une instruction INSERT, DELETE ou UPDATE porte sur la table (ou dans certains, cas sur la Vue associée).

Syntaxe

```
CREATE TRIGGER nom
BEFORE DELETE OR INSERT OR UPDATE ON
   tablename
[FOR EACH ROW WHEN (condition)]
DECLARE .....        <<<<déclarations>>>>
BEGIN
.....    <<<< bloc d'instructions PL/SQL>>>>
END;
```

Types de déclencheurs

- ORACLE propose deux types de triggers:
  1. **les triggers de table (STATEMENT)**
     - sont déclenchées une seule fois.
  2. **les triggers de ligne (ROW).**
     - se déclenchent individuellement pour chaque ligne de la table affectée par le trigger,
  - Si l'option FOR EACH ROW est spécifiée, c'est un trigger ligne, sinon c'est un trigger de table.
- doit être unique dans un même schéma
  - peut être le nom d'un autre objet (table, vue, procédure) mais à éviter

Option BEFOR/AFTER

- elle précise le moment quand ORACLE déclenche le trigger,
- les triggers AFTER row sont plus efficaces que les BEFORE row parce qu'ils ne nécessitent pas une double lecture des données.

Déclencheurs

- Elle comprend le type d'instruction SQL qui déclenche le trigger :
  - DELETE, INSERT, UPDATE
  - On peut en avoir une, deux ou les trois.
- Pour UPDATE, on peut spécifier une liste de colonnes. Dans ce cas, le trigger ne se déclenchera que si l'instruction UPDATE porte sur l'une au moins des colonnes précisée dans la liste.
  - S'il n'y a pas de liste, le trigger est déclenché pour toute instruction UPDATE portant sur la table.

les triggers lignes

- Pour les triggers lignes, on peut introduire une restriction sur les lignes à l'aide d'une expression logique SQL : c'est la clause WHEN :
  - Cette expression est évaluée pour chaque ligne affectée par le trigger.
  - Le trigger n'est déclenché sur une ligne que si l'expression WHEN est vérifiée pour cette ligne.
  - L'expression logique ne peut pas contenir une sous-question.
  - Par exemple, WHEN (:new.empno>0) empêchera l'exécution du trigger si la nouvelle valeur de EMPNO est 0, négative ou NULL.

Le corps du trigger

- Le corps du trigger est un bloc PL/SQL :
  - Il peut contenir du SQL et du PL/SQL.
  - Il est exécuté si l'instruction de déclenchement se produit et si la clause de restriction WHEN, le cas échéant, est évaluée à vrai.

Exemple1 de trigger table

```
CREATE TRIGGER log
AFTER INSERT OR UPDATE ON Emp
BEGIN
  INSERT INTO log (table, date, username, action)
  VALUES ('Emp', sysdate, sys_context('USERENV',
     'CURRENT_USER'), 'INSERT/UPDATE on Emp') ;
END ;
```

Exemple2 de trigger table

```
CREATE OR REPLACE
TRIGGER PERSON_UPDATE_SALAIRE
BEFORE UPDATE
ON Employe
BEGIN
DBMS_OUTPUT.PUT_LINE(' Avant la mise à jour de
   quelque employé');
END;
```

Exemple2 de trigger table

Maintenant, exécutant update...

UPDATE Employe SET sal= sal= sal+(sal*0.1);

SQL> UPDATE Employe SET sal= sal+(sal*0.1);

Avant la mise à jour de quelque employé

2 rows updated.

Les noms de corrélation (OLD/New)

- Lors de la création de triggers lignes, il est possible d'avoir accès à la valeur ancienne et la valeur nouvelle grâce aux mots clés OLD et NEW.
  - Il n'est pas possible d'avoir accès à ces valeurs dans les triggers de table.
- Si l'instruction de déclenchement du trigger est INSERT, seule la nouvelle valeur a un sens.
- Si l'instruction de déclenchement du trigger est DELETE, seule l'ancienne valeur a un sens.

Les noms de corrélation

- La nouvelle valeur est appelée :new.colonne
- L'ancienne valeur est appelée :old.colonne
  - Exemple : IF :new.salaire < :old.salaire then

Exemple1 de trigger row

```
CREATE OR REPLACE
TRIGGER  Employe_UPDATE_Salaire
BEFORE UPDATE
ON Employe
FOR EACH ROW
BEGIN
DBMS_OUTPUT.PUT_LINE('Avant la mise à jour ' ||
TO_CHAR(:OLD.sal) || '  vers ' || TO_CHAR(:NEW.sal));
END;
```

Exemple1 de trigger Row

Maintenant, exécutant update...

UPDATE Employe SET sal= sal= sal+(sal*0.5);

SQL> UPDATE Employe SET sal= sal+(sal*0.5);

Avant la mise à jour 1000 vers 1500

Avant la mise à jour 2000 vers 3000

Avant la mise à jour 4000 vers 6000

3 rows updated.

Exemple2 de trigger ligne

```
CREATE OR REPLACE TRIGGER difference_salaire
   BEFORE UPDATE ON Emp
   FOR EACH ROW
  WHEN (:new.Empno > 0)
  DECLARE
   sal_diff number;
  BEGIN
   sal_diff := :new.sal - :old.sal;
   dbms_output.put(' Old : ' || :old.sal || 'New : '
  || :new.sal || 'Difference : ' || sal_diff);
  END ;
```

```
drop SEQUENCE Employe$S1;
CREATE SEQUENCE Employe$S1  START WITH 1 MINVALUE 1 MAXVALUE 9999999999
    NOCACHE;
CREATE OR REPLACE TRIGGER Employe$T1
 BEFORE INSERT ON Employe
 FOR EACH ROW
BEGIN

        IF :New.ID IS  NULL THEN
                SELECT Employe$S1.NEXTVAL INTO :New.ID
                  FROM DUAL;
        END IF;

END;
/
```

L'option REFERENCING :

- Si une table s'appelle NEW ou OLD, on peut utiliser REFERENCING pour éviter l'ambiguïté entre le nom de la table et le nom de corrélation.
- Exemple :

```
CREATE TRIGGER nomtrigger
BEFORE UPDATE ON new
REFERENCING new AS newnew
FOR EACH ROW
BEGIN
:newnew.colon1:= TO_CHAR(:newnew.colon2);
END;
```

Les prédicats conditionnels INSERTING, DELETING et UPDATING

- Quand un trigger comporte plusieurs instructions de déclenchement (par exemple INSERT OR DELETE OR UPDATE), on peut utiliser des prédicats conditionnels (INSERTING, DELETING et UPDATING) pour exécuter des blocs de code spécifiques pour chaque instruction de déclenchement.

Les prédicats conditionnels INSERTING, DELETING et UPDATING

- Exemple :

```
CREATE TRIGGER ...
BEFORE INSERT OR UPDATE ON employe
.......
BEGIN
......
IF INSERTING THEN ..... END IF;
IF UPDATING THEN ..... END IF;
......
END;
```

Les prédicats conditionnels INSERTING, DELETING et UPDATING

- UPDATING peut être suivi d'un nom de colonne :

```
CREATE TRIGGER ...
BEFORE UPDATE OF salaire, commission ON employe
.......
BEGIN
......
IF UPDATING ('salaire') THEN ........ END IF;
......
END;
```

Nombre de triggers par table

- On peut avoir au maximum un trigger de chacun des types suivants pour chaque table :

BEFORE UPDATE row
BEFORE DELETE row
BEFORE INSERT row
    BEFORE INSERT statement
    BEFORE UPDATE statement
    BEFORE DELETE statement
AFTER UPDATE row
AFTER DELETE row
AFTER INSERT row
    AFTER INSERT statement
    AFTER UPDATE statement
    AFTER DELETE statement.

Nombre de triggers par table

- Même pour UPDATE, on ne peut pas en avoir plusieurs avec des noms de colonnes différents.

Instructions SQL autorisées

- les instructions du LMD sont autorisées
- les instructions du LDD ne sont pas autorisées
- les instructions de contrôle de transaction (ROLLBACK, COMMIT) ne sont pas autorisées.

Triggers en cascade

- Un trigger peut provoquer le déclenchement d'un autre trigger.
- ORACLE autorise jusqu'à 32 triggers en cascade à un moment donné.

Conditions nécessaires pour créer un trigger

- il faut avoir le privilège CREATE TRIGGER
- il faut soit posséder la table sur laquelle on veut définir un trigger, soit posséder le privilège ALTER sur la table sur laquelle on veut définir le trigger, soit posséder le privilège ALTER ANY TABLE

Modification de triggers

- Pour modifier un trigger, on refait une instruction CREATE TRIGGER suivie de OR REPLACE ou bien on supprime le trigger (DROP TRIGGER nomtrigger) et on le crée à nouveau.

Activation d'un trigger

- Un trigger peut être activé ou désactivé.
  - Par défaut, un trigger est activé dès sa création.
- Pour désactiver un trigger, on utilise l'instruction
  - ALTER TRIGGER avec l'option DISABLE :
  - ALTER TRIGGER nomtrigger DISABLE;
- On peut désactiver tous les triggers associés à une table avec la commande :
  - ALTER TABLE nomtable DISABLE ALL TRIGGERS;
- A l'inverse on peut réactiver un trigger :
  - ALTER TRIGGER nomtrigger ENABLE;
- ou tous les triggers associés à une table :
  - ALTER TABLE nomtable ENABLE ALL TRIGGERS;

vues du dictionnaire (Triggers)

- Les définitions des triggers sont stockées dans les tables de la métabase, notamment dans les tables USER_TRIGGERS, ALL_TRIGGERS et DBA_TRIGGERS

ADMINISTRATION DU SYSTÈME ORACLE 10G

(IGE Semestre 5 Faculté Polydiscilinaire de Ouarzazate)

COMPOSITION DU MODULE

408

Matières

1 : Architecture et Installation Oracle

2 : Programmation sous oracle (PL/SQL)

3 : Les outils d'administration Oracle

409

Gestion de la sécurité

Qu'est-ce que la gestion de la sécurité ?

410

- Problèmes de sécurité :si les données sont « sensibles » elles doivent être protégées des accès « frauduleux »
- Sécurité = protection des données
  - Piratage
  - Accès non autorisés
  - Manipulations non autorisées
  - …

Qu'est-ce que la gestion de la sécurité ?

411

- Gestion de la sécurité sous Oracle
  - Plusieurs niveaux de sécurité :
    - Niveau Oracle
      - Utilisateurs
      - Rôles, privilège
      - Profiles
    - Niveau Système d'exploitation
      - Autorisation d'accès au serveur
      - Protection des fichiers Oracle
        - fichiers de données, fichiers de reprise, fichiers de contrôle, dump, sauvegardes, …
- Possibilité d'audit pour chaque niveau de sécurité

412

Gestion des utilisateurs

Schéma de base de données

413

- Pour pouvoir accéder aux données, on doit se connecter via un compte utilisateur qui aura certains privilèges et une certaine visibilité de la base de données.
- Il existe 2 utilisateurs par défaut sur toute base Oracle :
  - l'utilisateur SYS, propriétaire des tables et des vues du dictionnaire
  - l'utilisateur SYSTEM, qui a simplement le droit de consultation de ces objets.
  - Ces 2 utilisateurs ont par défaut le rôle DBA, ce qui veut dire qu'ils ont accès à tous les objets de tous les autres utilisateurs de la base, et qu'ils ont le droit d'exécuter certaines commandes d'exploitation et d'administration.
- Un DBA peut créer des utilisateurs en utilisant la requête CREATE USER.
  - Lorsqu'un utilisateur est créé, il ne possède aucun privilège. Le DBA doit lui donner des privilèges souhaités.

Schéma de base de données

414

- A chaque création d'un utilisateur correspond la création d'un schéma de même nom
- Un compte utilisateur = Un schéma de BDD
  - C'est un ensemble de d'objets : tables, vues, index,...
  - L'utilisateur crée, modifie,... ses objets

Etapes de création d'un utilisateur

415

1. Choisir un nom d'utilisateur et un mécanisme d'authentification
   - par Oracle
   - Par le SE
2. Identifier les tablepaces dans lesquels l'utilisateur va stocker ses objets
   - typiquement 3 tablespaces pour les données, les tris et les index
     - Revient à affecter un tablespace par défaut et un tablespace temporaire
3. Décider des quotas pour chaque tablespace
4. Créer l'utilisateur
5. Accorder des privilèges et des rôles à l'utilisateur

Exemples

416

CREATE USER Toto

IDENTIFIED [BY xyz /EXTERNALLY]

DEFAULT TABLESPACE data01

TEMPORARY TABLESPACE temp

QUOTA 15m on data01 [UNLIMITED]

PROFILE Profil1

PASSWORD EXPIRE

ACCOUNT UNLOK [LOCK];

Modification

417

- Avec ordre ALTER USER
  - force changement de mot de passe

    ALTER USER Toto
    PASSWORD EXPIRE;
  - Suppression de quota

    ALTER USER Toto
    QUOTA 0 ON data01;
    - les données existantes restent
    - mais plus possible d'en insérer d'autres
  - quota illimité

    ALTER USER Toto
    QUOTA UNLIMITED ON data01;
  - interdiction temporaire de connexion

    ALTER USER Toto
    ACCOUNT LOCK ;

Suppression

418

- Syntaxe :
  - DROP USER utilisateur [CASCADE]
- Option CASCADE
  - supprime tous les objets du schéma
  - puis supprime l'utilisateur
- Exemple :
  - DROP USER Toto CASCADE;
- Remarque :
  - impossible de supprimer un utilisateur connecté

Informations sur les utilisateurs

419

- Utiliser les vues
  - DBA_USERS
  - DBA_TS_QUOTAS
- Exemples :

```
Select username,default_tablespace, temporary_tablespace
from dba_users
where username = 'Toto';

select *  from dba_ts_quotas
where username= 'Toto';
```

420

Gestion des profils

Profils

421

- Ensembles nommés de limites de ressource
  - nombre de sessions (connexion) simultanées par utilisateur
  - durée d'inactivité
  - durée totale de connexion
  - durée de vie du mot de passe (jours)
  - …
- Affectés aux utilisateurs
  - lors de leur création
  - par modification
- Peuvent être activés/désactivés
- Sont utilisés sur les gros systèmes pour contrôler l'utilisation des ressources d'un groupe d'utilisateur par rapport à un autre

Gestion des ressources à l'aide des profils

422

- Etapes à suivre :
  - créer les profils
    - ordre CREATE PROFIL
  - les affecter à l'utilisateur
    - ordre CREATE/ALTER USER
  - activer les limites de ressources
    - soit ALTER SYSTEM
      SET RESOURCE_LIMIT =TRUE or FALSE
    - soit fichier de paramètres d'initialisation

Gestion des ressources à l'aide des profils

423

```
CREATE PROFILE developer_prod  LIMIT
SESSIONS_PER_USER 2 [UNLIMITED / DEFAULT]
 // nb sessions simultanées
IDLE TIME 60 [UNLIMITED / DEFAULT]
// durée d'inactivité
CONNECT TIME 480 [UNLIMITED / DEFAULT]
// durée totale de connexion…
PASSWORD_LIFE_TIME 30 [UNLIMITED / DEFAULT]
// durée de vie du mot de passe (jours);

ALTER USER Toto
 PROFILE developer_prod;
```

Modification/suppression

424

- Modification
  - ordre ALTER PROFILE
  - Exemple :
    - alter profile developer_prod limit
      sessions_per_user 3
      idle_time 2;
- Suppression
  - ordre DROP PROFILE
    - option CASCADE : assure que tous les utilisateurs ayant ce profil seront mis à jour!
  - le profile DEFAULT ne peut être supprimé
  - Exemple :
    - DROP PROFILE developer_prod CASCADE;
- Entre en vigueur pour les sessions suivantes

Informations sur les profils

425

- Utiliser les vues
  - DBA_USERS
  - DBA_PROFILES
- Exemples :

```
select distinct profile
    from dba_profiles;
select *
    from dba_profiles
    where profile='DEFAULT';
```

426

Gestion des privilèges

Privilèges

427

- Les privilèges sont des droits pour exécuter des requêtes
  - Deux types de Privilèges
    - SYSTEME
      - permet aux utilisateurs d'effectuer des opérations touchant la structure de la base (create tablespace, create database, alter system , create table, create user, alter profile …)
      - Il existe a peu prés 127 privilèges système qui sont classés par catégories
    - OBJET
      - Un privilège objet permet d'exécuter une action particulière sur une table, vue, fonction, séquence, procédure, package d'un schéma.
        - Par exemple y accéder, la mettre a jour ou même y insérer des information (select, update, insert , references, execute…)

Privilèges

428

- Ordre GRANT permet d'ajouter un privilège à un utilisateur
  - Exemple :
    - GRANT create index, create table TO Toto;
- Ordre REVOKE pour le supprimer
  - Exemple :
    - REVOKE create table FROM Toto;

Attribution de privilèges systèmes

429

- Avec l'option
  - GRANT … TO …
    WITH ADMIN OPTION;
- Pour accorder un privilège système, il faut posséder le privilège WITH ADMIN OPTION

Attribution de privilèges systèmes

430

- Privilèges SYSDBA et SYSOPER (AS SYSDBA / AS SYSOPER)
  - SYSOPER (startup, shutdown, archive log, recover, Alter database, open/mount et alter database backup, restricted session)
  - SYSDBA = tous les privilège systèmes avec (WITH ADMIN OPTION) , tous les privilège SYSOPER + CREATE DATABASE et de faire de recouverement basé sur le temps

Privilèges Objets

431

- Les principaux privilèges Objet :
  - ALTER, DELETE, INDEX, REFERENCES SELECT, UPDATE (pour les tables), EXECUTE (pour les procédures)
- Exemples :
  - GRANT execute ON dbms_pipe TO Toto;
  - GRANT select, update (nom,sal) ON emp TO Bob WITH GRANT OPTION;
  - REVOKE execute ON dbms_pipe FROM scott;
- Attention à la formulation :
  - pour donner le privilège "with admin option", il faut faire :
    - GRANT … WITH GRANT OPTION

Informations sur les privilèges

- Interroger les vues
  - pour les privilèges SYSTEM :
    - DBA_SYS_PRIVS
    - SESSION_PRIVS
  - pour les privilèges OBJET :
    - DBA_TAB_PRIVS
    - DBA_COL_PRIVS
- Exemple :
  - select * from DBA_SYS_PRIVS;
  - select * from SESSION_PRIVS;
  - select * from DBA_TAB_PRIVS;

Gestion des rôles

Rôles

434

- Oracle défini des « rôles » comme un ensemble de privilèges.
  - On utilisera/définira les rôles qui conviennent aux différents type de besoins.
  - Oracle prédéfini des rôles, par exemple :
    - Connect : peut se connecter à la base et créer des objets de base,
    - Resource : peut créer de code PL/SQL stocké dans la base,
    - DBA : tous les privilèges système avec l'option ADMIN
    - IMP_FULL_DATABASE, EXP_FULL_DATABASE, …

Création/Affectation de Rôles

435

- Création
  - Ordre CREATE ROLE
  - Exemples :
    - CREATE ROLE Vente;
    - CREATE ROLE Vente IDENTIFIED BY bonus;
    - CREATE ROLE Vente IDENTIFIED EXTERNALLY;
- Affectation
  - Ordre GRANT
  - Exemples :
    - GRANT create any table TO Vente ;
    - GRANT Vente TO scott;
    - GRANT Vente TO scott WITH ADMIN OPTION;

Rôles par défaut aux utilisateurs

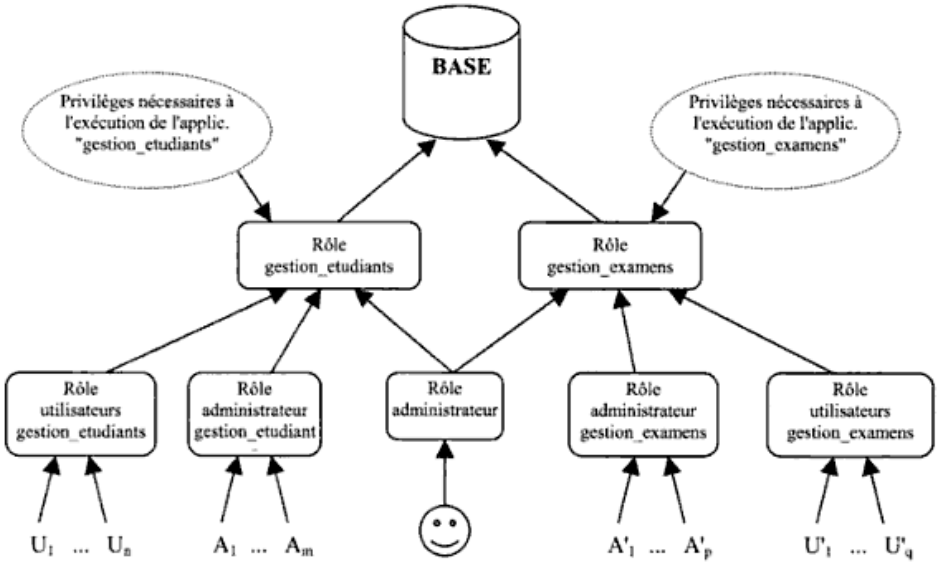
436

- Exemples :
  - ALTER USER scott
    DEFAULT ROLE Vente, Achat;
  - ALTER USER scott
    DEFAULT ROLE ALL;
  - ALTER USER scott
    DEFAULT ROLE ALL EXCEPT Vente;
  - ALTER USER scott
    DEFAULT ROLE NONE;

Activation on/off des rôles

437

- Relatif à la session
  - intéressant dans une application accédant la base pour s'assurer que la personne qui exécute l'application est bien autorisée.
- Commande SET ROLE (pour activer un rôle)
  - Exemples :
    - SET ROLE Vente;
    - SET ROLE Vente IDENTIFIED BY commission;
    - SET ROLE ALL EXCEPT Achat;
    - SET ROLE NONE;
- Suppression des rôles :
  - DROP ROLE role
- Suppression des rôles d'un utilisateur
  - REVOKE role FROM Toto;



438

Un rôle est créé au moyen de la commande **CREATE ROLE**. Par exemple :

```
CREATE ROLE app_gestion_etudiants;
```

Lorsqu'un rôle a été créé, on peut lui associer des privilèges des rôles préalablement définis en utilisant la commande **GRANT**.

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON eleves TO app_gestion_etudiants;
GRANT SELECT, INSERT, UPDATE, DELETE
ON activites TO app_gestion_etudiants;
GRANT SELECT, INSERT, UPDATE, DELETE
ON activites_pratiquees TO app_gestion_etudiants;
```

Si on crée le rôle `utilisateur_gestion_etudiants` :

439

On peut lui associer le rôle applic_gestion_etudiants défini ci-dessus et le privilège système CREATE SESSION autorisant une connexion à la base :

```
GRANT app_gestion_etudiants, CREATE SESSION
   TO utilisateur_gestion_etudiants;
```

Il reste, enfin, à accorder ce rôle pour les différents utilisateurs :

```
GRANT utilisateur_gestion_etudiants
   TO titine, sualem, betty;
```

Avant qu'un utilisateur ne puisse faire usage d'un rôle qu'il vient de recevoir, il doit l'activer. Ceci se fait au moyen de la commande **SET ROLE** *liste_roles*. Par exemple :

```
SET ROLE utilisateur_gestion_etudiants;
```

Les rôles ne figurant pas dans la *liste_roles* sont désactivés. Pour remédier à cela, on peut utiliser **SET ROLE ALL** pour activer tous les rôles dont on dispose. Il est également possible de désactiver un rôle au moyen de l'option **EXCEPT** :

Exemple

- Une application de gestion de stock défini trois tables CLIENTS, STOCK, PRODUITS et une vue PRODUITS_EN_STOCK.
  - Un utilisateur « GSTOCK » est défini et est propriétaire des tables et de la vue :
    - Il doit pouvoir créer des tables et des vues
    - Les objets seront créées par lui.
  - Un rôle de consultation des données est créé :

    CREATE ROLE CONSULT_DATA ;

    Les privilèges correspondant lui sont affectés :

    GRANT SELECT ON GSTOCK.CLIENT TO CONSULT_DATA; … (un grant par objet, mais plusieurs privilèges possible par objet)…
  - Un rôle de mise à jours du stock est créé :

    CREATE ROLE UPDT_STOCK ;

    Les privilèges correspondant lui sont affectés :

    GRANT SELECT, UPDATE ON GSTOCK.STOCK TO UPDT_STOCK ;

    GRANT SELECT ON GSTOCK.CLIENT TO UPDT_STOCK ;

    GRANT SELECT ON GSTOCK.PRODUITS TO UPDT_STOCK ;

    GRANT UPDATE(NbIn, NbOut) ON GSTOCK.PRODUITS TO UPDT_STOCK ;
  - Un utilisateur WEBSTOCK de l'application Web de Mise à jours du stock est créé, le rôle UPDT_STOCK sera son rôle par défaut :

    GRANT UPDT_STOCK TO WEBSTOCK ;

    ALTER USER WEBSTOCK DEFAULT ROLE UPDT_STOCK ;

Conseils

442

- Créer un rôle pour chaque tâche d'application
- Créer un rôle pour chaque type d'utilisateur
- Accorder des rôles d'utilisateurs aux utilisateurs

Informations sur les rôles

443

- Interroger les vues
  - DBA_ROLES
    - tous les rôles de la bd
  - DBA_ROLE_PRIVS
    - rôles accordés aux utilisateurs et aux rôles
  - ROLE _ROLE_PRIVS
    - rôles accordés aux rôles
  - DBA_SYS_PRIVS
    - Privilèges accordés aux utilisateurs et aux rôles
  - ROLE_SYS_PRIVS
    - Privilèges systèmes accordés aux rôles
  - ROLE_TAB_PRIVS
    - Privilèges de table accordés aux rôles
  - SESSION_ROLES
    - rôles d'un utilisateur actuellement activés

444

L'audit sous Oracle

AUDIT

445

- Auditer la BDD, c'est chercher les activités suspectes et inappropriées.
  - L'audit Oracle est une des possibilités de surveillance de l'activité de la base de données :
  - pour contrôler les accès à la base, à des fins de sécurité,
  - pour vérifier que tel ou tel objet est accédé en lecture ou en écriture (sécurité ou analyse de performance),
  - pour vérifier les tentatives d'accès infructueux à des objets,

L'audit sous Oracle

446

- Le déclenchement de l'audit effectif se fait par **la commande AUDIT**
  - Les résultats de l'audit seront Enregistrés dans la table SYS.AUD$, ou dans le SE
    - la table SYS.AUD$ est crée par le script CATAUDIT.sql
  - Pour pouvoir auditer la base de données il faut mettre le paramètre AUDIT TRAIL soit BD, soit OS
    - NONE : invalide l'audit (valeur par default)
    - DB : valide l'audit et stocke les résultats dans la table d'audit
    - OS : valide l'audit et stocke les résultats dans un fichier externe (un autre paramètre : AUDIT_FILE_DEST précise le répertoire de destination...)

L'audit sous Oracle

447

- A chaque niveau d'audit, on peut de + surveiller aussi bien LES SUCCES que LES ECHECS
- la table SYS.AUD$ et les vues DBA_% ne sont bien accessibles qu'au par DBA
- la table SYS.AUD$ doit être surveillée et purgée par un DELETE (ou mieux un TRUNCATE) explicite de SYS si nécessaire

Audit sur les commandes et les privilèges

448

- AUDIT privilège,commande [privilège,commande,…]
  [By utilisateur [utilisateur,…]
  [By {access/session}]
  [whenever [not] successful]

L'audit sous Oracle

449

- Audit de login
  - AUDIT SESSION WHENEVER NOT SUCCESSFULL

  surveille toutes les tentatives de connexions infructueuses
- AUDIT CREATE ANY PROCEDURE
  BY ACCESS WHENEVER SUCCESSFULL
  audit système : surveille les création de procédures réussies sur toutes la base

L'audit sur les schémas d'objets

450

- AUDIT SELECT TABLE, DELETE TABLE
  BY SESSION
  WHENEVER NOT SUCCESSFUL
  surveille les select et insert infructueux sur n'importe quelle table, une entrée seulement par session
- AUDIT INSERT ON scott.dept
  audit objet : surveille les insertion (réussies ou échouées) sur la table DEPT de SCOTT

L'audit sous oracle

451

- **Sélection des résultats dans les vues d'audit**
  - Faire un SELECT sur DBA_AUDIT_OBJECT ou DBA_AUDIT_SESSION ou DBA_AUDIT_STATEMENT

452

Sauvegarde / restauration

Problèmes liés à la sauvegarde et à la récupération

453

- Le DBA doit:
  - Protéger la base de données contre de nombreux types d'incident
  - Augmenter la durée moyenne sans pannes
  - Réduire la durée moyenne de récupération
  - Minimiser la perte de données

Exemples d'incidents

454

- Une défaillance peut être causée par exemple par l'un des événements suivants :
  - Utilisation d'une clause UPDATE dans un environnement de production, en oubliant la clause WHERE.
  - Défaillance d'un disque dur supportant la base de données.
  - Corruption d'un fichier de données.
  - Destruction physique d'un ordinateur causée par les éléments naturels (feu, inondation).
  - …

Définir une stratégie de sauvegarde

455

- Une analyse est requise pour définir la stratégie de sauvegarde :
  - Est-il acceptable de perdre des données ?
  - Est-il possible d'arrêter périodiquement la base de données ?
  - Est-il possible de réaliser une sauvegarde complète de la base de données pendant l'arrêt ? Dans des temps raisonnables ?
  - Les données sont-elles mises à jour quotidiennement/périodiquement par les utilisateurs ?
  - Quel est le budget alloué pour les sauvegardes ? (La très haute disponibilité demande des coûts importants).

Définir une stratégie de sauvegarde

456

- Une sauvegarde sera nécessaire dans les cas suivants :
  - Après une modification de la structure physique de la base de données (ajout ou suppression d'un tablespace, ajout d'un fichier de données, ajout ou suppression d'un fichier de journalisation).
  - Après le déplacement d'un fichier de données ou d'un fichier de journalisation.
  - Après un chargement volumineux de données (surtout avec SQL* Loader).
  - Après la modification d'un volume important de données.

Définir une stratégie de sauvegarde

457

- Le mode **ARCHIVELOG** permet de garantir zéro perte de données en cas d'incident sur un fichier de données.
  - **N.B** : En réalisant des sauvegardes fréquentes, vous vous assurerez une restauration en temps raisonnable. En effet, vous devrez appliquer un nombre minimum de fichiers de journalisation.
  - Les bases de données Oracle sont par défaut créées en mode **NOARCHIVELOG**

Modifier le mode d'archivage

1. Modifier le fichier de paramètre (pfile).
2. Eteindre la base de données.
3. Lancer la base de données en mode **MOUNT**.
4. Redémarrer la base de données.

-- pour automatiser l'archivage

- LOG_ARCHIVE_START=True
- SHUTDOWN IMMEDIATE;
- STARTUP MOUNT ;
- ALTER DATABASE ARCHIVELOG;
- ALTER DATABASE OPEN;

Solutions de sauvegarde et restauration

459

- Deux possibilités s'offrent à vous pour effectuer des sauvegardes et des restaurations :
  - Utiliser l'outil Recovery Manager (RMAN). C'est un outil ligne de commande qui est fourni par Oracle. Il facilite les opérations de sauvegarde et de restauration, en limitant les risques de fausses manipulations.
  - Procéder à la main en utilisant des commandes système et des scripts SQL.

Stratégies de sauvegarde

460

- Si vous désirez sauvegarder manuellement une base de données, deux possibilités s'offrent à vous :
  - les sauvegardes à chaud (BDD Ouverte)
  - les sauvegardes à froid (BDD arrêtée)

Sauvegarde à froid

461

- La plus simple à mettre en œuvre
- Les étapes :
  - identification des fichiers
    1. select name from v$datafile;
    2. select member from v$logfile;
    3. select name from v$controlfile;
    4. select name from v$tempfile;
  - arrêter la base de données
  - sauvegarder tous ces fichiers
    - commandes du SE
  - Redémarrer la base
- En cas d'erreur, copie de ces fichiers pour redémarrer
  - perte d'information possible
  - A privilégier si la BD n'est pas dans un environnement transactionnel élevé
- Notez qu'une sauvegarde à froid peut-être effectuée quand la base de données est en mode **NOARCHIVELOG.**

Sauvegarde à froid

462

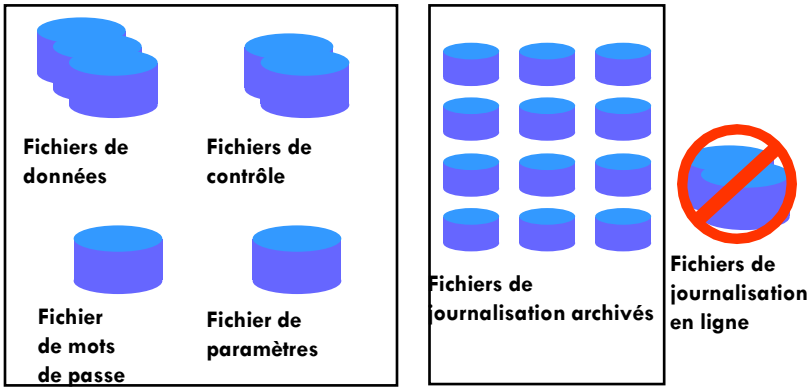
```
-- Variables d'environnement de SQL*Plus de formatage de l'affichage
Set feedback off
Set Linesize 200
Set Heading off
Set Pagesize 0
Set Trimspool off
Set Verify off
define repertoire ='c:\archive' --  répertoire de destination des fichiers sauvegardés
define fichier_control=c:\control_backup.sql  -- définition du fichier de sortie
spool &fichier_control
select 'host copy  ' || name || ' &repertoire ' from v$datafile order by 1 ;
select 'host copy  ' || member || ' &repertoire ' from v$logfile order by 1 ;
select 'host copy  ' || name  || ' &repertoire ' from v$controlfile order by 1 ;
select 'host copy  ' || name  || ' &repertoire ' from v$tempfile order by 1 ;
spool off
-- Fermeture de la base de données pour avoir des fichiers synchronisés
shutdown immediate
@&fichier_control
startup
```

Sauvegarde à chaud

463

- Une sauvegarde à chaud « Hot backup » est une sauvegarde effectuée lorsque la base de données est ouverte.
  - L'activité se poursuit pendant la sauvegarde.
  - Lorsque la base de données est restaurée, il faut appliquer les fichiers de journalisation pour rendre la base cohérente.
  - Ce type de sauvegarde n'est possible que si la base de données fonctionne en mode **ARCHIVELOG**

Sauvegarde à chaud



Sauvegarde à chaud


```
SET feedback off pagesize 0 heading off verify off linesize 100 trimspool on
PROMPT veuillez entrer le chemin du répertoire destinataire des sauvegardes
ACCEPT repertoire
PROMPT veuillez entrer le chemin du premier fichier
ACCEPT fichier
PROMPT veuillez entrer le chemin du second fichier
ACCEPT spool
SPOOL &fichier
PROMPT spool &spool  ;;
PROMPT archive log list ;;
PROMPT alter system switch logfile ;;
SELECT ' alter tablespace ' || tablespace_name || ' begin backup  ; '
  FROM dba_tablespaces
 WHERE status NOT IN ('READ ONLY', 'INVALID', 'OFFLINE');
SELECT ' host copy ' || file_name || ' &repertoire '
  FROM dba_data_files
 WHERE tablespace_name NOT IN (
                          SELECT tablespace_name
                            FROM dba_tablespaces
                           WHERE status IN
                                      ('READ ONLY', 'INVALID', 'OFFLINE'));
SELECT ' alter tablespace ' || tablespace_name || ' end backup  ; '
  FROM dba_tablespaces
 WHERE status NOT IN ('READ ONLY', 'INVALID', 'OFFLINE');
PROMPT alter database backup controlfile to '&repertoire\control.ctl' REUSE ;;
PROMPT alter system switch logfile ;;
PROMPT archive log list ;;
PROMPT spool off ;;
SPOOL off;
@&fichier
```

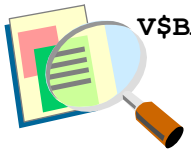
Sauvegarde à chaud

466

```
spool c:\oracle\sauvegarde\hot_backup.sql ;
archive log list ;
alter system switch logfile ;
 alter tablespace SYSTEM begin backup  ;
 alter tablespace RBS begin backup  ;
 alter tablespace USERS begin backup  ;
 alter tablespace TEMP begin backup  ;
 alter tablespace TOOLS begin backup  ;
 alter tablespace INDX begin backup  ;
 alter tablespace DRSYS begin backup  ;
 host copy C:\ORACLE\ORADATA\BDO\USERS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\DR01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\TOOLS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\INDX01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\RBS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\TEMP01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BDO\SYSTEM01.DBF c:\oracle\sauvegarde
 alter tablespace SYSTEM end backup  ;
 alter tablespace RBS end backup  ;
 alter tablespace USERS end backup  ;
 alter tablespace TEMP end backup  ;
 alter tablespace TOOLS end backup  ;
 alter tablespace INDX end backup  ;
 alter tablespace DRSYS end backup  ;
alter database backup controlfile to 'c:\oracle\sauvegarde\control.ctl' REUSE ;
alter system switch logfile ;
archive log list ;
spool off ;
```

Informations sur l'état d'une sauvegarde

Vues dynamiques



V$BACKUP

V$DATAFILE_HEADER

469

Restaurations

Récupération en mode NOARCHIVELOG

- En mode NOARCHIVELOG, vous devez restaurer les fichiers de base de données suivants :
  - tous les fichiers de données
  - les fichiers de contrôle
- Vous pouvez également restaurer :
  - les fichiers de journalisation
  - le fichier de mots de passe
  - le fichier de paramètres

Récupération en mode NOARCHIVELOG

- Avantages
  - Simplicité d'exécution et faible risque d'erreur
  - La durée de la récupération correspond à la durée de restauration de tous les fichiers
- Inconvénients
  - Les données sont perdues et doivent être réappliquées manuellement
  - La base de données entière est restaurée jusqu'au point de la dernière sauvegarde totale base fermée

Restauration d'une base de données en mode NOARCHIVELOG

1. Arrêtez l'instance.
2. Restaurez l'ensemble de fichiers à partir de la dernière sauvegarde totale de la base de données.
3. Déplacer le fichier de contrôle vers son nouvel emplacement, puis renseigner la valeur du paramètre CONTROL_FILES dans le fichier d'initialisation du nouvel emplacement
4. Démarrer l'instance et monter la base de données pour lire le fichier de contrôle, et qu'il soit ainsi accessible.
5. Modifier le fichier de contrôle pour qu'il contienne les nouveaux emplacements des fichiers de données
   - Exemple: alter database rename 'c:\oracle\oradata\data01.dbf' to 'd :\oracle\oradata\data01.dbf';
6. Ouvrir la BDD

Récupération en mode ARCHIVELOG

- Récupération complète
  - met à jour la base de données au point le plus récent
  - utilise des données de journalisation
  - applique toutes les modifications journalisées
- Récupération incomplète
  - met à jour la base de données au point donnée avant la défaillance de la BDD
    - Jusqu'un SCN (system change number)
    - À une date et heure
    - Basé sur l'annulation
  - Applique les archive log

Récupération complète en mode ARCHIVELOG

- Avantages
  - Seuls les fichiers perdus doivent être restaurés
  - Toutes les données sont récupérées, jusqu'au moment de la défaillance
  - Le temps de récupération correspond à la durée nécessaire pour restaurer les fichiers perdus et appliquer tous les fichiers de journalisation archivés
- Inconvénients
  - Vous devez disposer de tous les fichiers de journalisation archivés depuis la sauvegarde utilisée pour la restauration

Récupération et restauration physique de fichiers de données à l'aide de procédures gérées par l'utilisateur

- Restaurez les fichiers à l'aide de commandes du système d'exploitation
- Récupérez les fichiers à l'aide de la commande SQL*Plus `RECOVER`

487

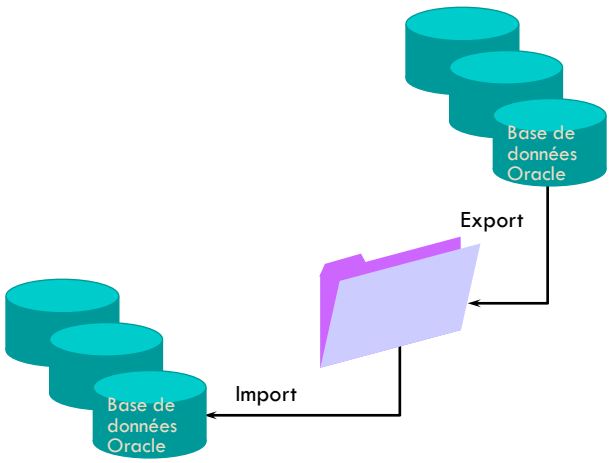
Les Utilitaires EXPORT et IMPORT

Présentation des utilitaires Export et Import d'Oracle

Ces utilitaires permettent d'effectuer les opérations suivantes :

- Archiver des données d'historique
- Enregistrer des définitions de tables pour les protéger de tout incident utilisateur
- Déplacer des données entre des ordinateurs et des bases de données, ou entre différentes versions du serveur Oracle
- Transférer des tablespaces entre des bases de données

Présentation des utilitaires Export et Import d'Oracle



Présentation des utilitaires Export et Import d'Oracle

490

- L'utilitaire Export peut fournir une sauvegarde logique :
  - Des tables
  - Des utilisateurs
  - des tablespaces
  - de la base de données entière
- L'utilitaire Import permet de lire un fichier d'export valide pour déplacer des données dans une base de données.
  - L'historique des fichiers de journalisation (redo log history) ne peut pas s'appliquer à des objets importés à partir d'un fichier d'export ; par conséquent, une perte de données peut survenir, mais celle-ci peut être minimisée.

Présentation des utilitaires Export et Import d'Oracle

491

- Vous pouvez exploiter les utilitaires Export et Import en complément des sauvegardes ordinaires effectuées via le système d'exploitation, en les utilisant pour :
  - créer une archive historique d'un objet de base de données ou d'une base de données entière, par exemple, lorsqu'un schéma est modifié pour tenir compte de nouveaux impératifs au sein de l'entreprise
  - enregistrer des définitions de tables dans un fichier binaire, ce qui peut s'avérer utile lors de la création et de la gestion d'une ligne de base pour une structure de schéma donnée
  - déplacer des données d'une version de base de données Oracle à une autre, par exemple lors d'une mise à niveau d'Oracle9*i* vers Oracle10g

Présentation des utilitaires Export et Import d'Oracle

492

- Ces utilitaires assurent une protection dans le cas des erreurs suivantes :
  - les erreurs utilisateur (suppression ou vidage accidentel d'une table)
  - l'endommagement logique d'une table
  - un traitement batch incorrect ou une instruction LMD qui n'a affecté qu'un sous-ensemble de la base de données

Présentation des utilitaires Export et Import d'Oracle

493

- L'utilitaire Export crée un fichier contenant une copie logique de tout ou une partie d'une base de données oracle
- L'utilitaire Export fournit un moyen simple de transférer des objets de données entre des bases Oracle même si celles-ci figurent sur des plates-formes dotées de configurations matérielles et logicielles différentes.

Présentation des utilitaires Export et Import d'Oracle

494

- Lorsque vous exécutez l'utilitaire Export sur une base de données, des objets (tels que des tables) sont extraits, suivis par les objets qui leur sont associés (par exemple, les index, les commentaires et les privilèges), le cas échéant. Les données extraites sont écrites dans un fichier d'export, qui est un fichier dump Oracle de format binaire généralement situé sur disque ou sur bande.
- L'utilitaire Import lit les définitions d'objet et les données de table à partir du fichier dump d'Export, puis il insère les objets de données dans une base Oracle.

Méthodes d'appel des utilitaires Export et Import

- Interface de ligne de commande
- Fichier de paramètres
- Oracle Enterprise Manager

Modes d'export

Mode "Table"	Mode "User"	Mode "Tablespace"	Mode "Full"
Définitions de tables	Définitions de tables	Définitions de tables	Définitions de tables
Données de table (toutes les lignes ou les lignes sélectionnées)	Données de tables	Privilèges	Données de tables
Privilèges du propriétaire sur les tables	Privilèges du propriétaire	Index	Privilèges
Index de table du propriétaire	Index du propriétaire	Contraintes de table	Index Contraintes de table
Contraintes de table	Contraintes de table	Déclencheurs	**(à l'exception du schéma SYS)**

Appeler l'utilitaire Export

- Syntaxe :

```
exp utilisateur/mot de passe [paramètres]
```

- Exemples :

```
exp hr/hr TABLES=employees,departments
    rows=y file=exp1.dmp
```

```
exp system/manager OWNER=hr direct=y
    file=expdat.dmp
```

498

- Pour utiliser Export, vous devez disposer du privilège `CREATE SESSION` sur une base de données Oracle.
- Pour exporter des tables appartenant à un autre utilisateur, il est nécessaire que le rôle `EXP_FULL_DATABASE` soit activé pour vous. Ce rôle est accordé à tous les DBA.
  - Si vous ne détenez pas les privilèges système du rôle `EXP_FULL_DATABASE`, vous ne pouvez pas exporter des objets contenus dans le schéma d'un autre utilisateur.

499

- Pour utiliser Import, vous avez besoin du privilège `CREATE SESSION` pour vous connecter au serveur de bases de données Oracle.
  - Ce privilège fait partie du rôle `CONNECT` établi lors de la création de la base de données.
- Vous pouvez effectuer un import même si vous n'avez pas créé le fichier d'export. Toutefois, si ce dernier a été créé par un autre utilisateur, vous ne pourrez l'importer que si vous disposez du rôle `IMP_FULL_DATABASE`.

Modes d'import

Mode	Description
Table	Importer les tables indiquées dans un schéma.
User	Importer tous les objets appartenant à un schéma
Tablespace	Importer toutes les définitions des objets du tablespace
Full Database	Importer tous les objets à partir du fichier d'export

Appeler l'utilitaire Import

- Syntaxe :

```
imp user/password [parameters]
```

- Exemples :

```
imp hr/hr TABLES=employees,departments
    rows=y file=exp1.dmp
```

```
imp system/manager FROMUSER=hr file=exp2.dmp
```

Séquence du processus d'import

1. Les nouvelles tables sont créées
2. Les données sont importées
3. Les index sont construits
4. Les déclencheurs sont importés
5. Les contraintes d'intégrité sont activées sur les nouvelles tables
6. Tous les index bitmap, fonctionnels et/ou de domaine sont construits

503

*Chargement de Données avec SQL*Loader*

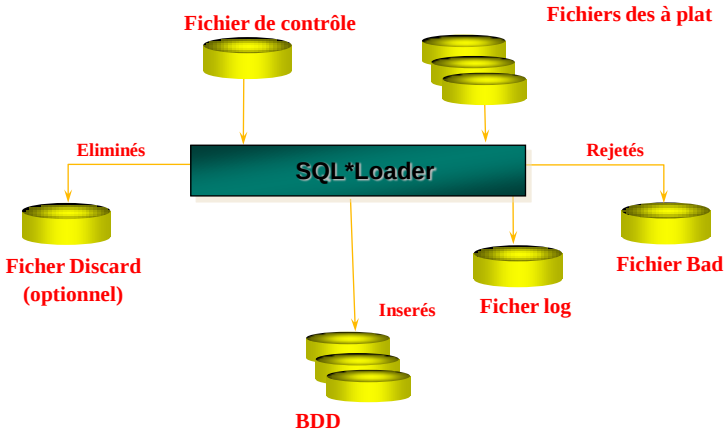
SQL*Loader

- SQL*Loader sert à insérer (charger) les données dans
  - des tables d'une base de données oracle à partir d'un fichier ASCII.
- C'est un utilitaire souple qui permet de choisir les colonnes de la table à charger et de sélectionner les enregistrements.

SQL*Loader

- Pour charger les données dans les tables de la base Oracle, il faut construire:
  - Un fichier de contrôle qui décrit comment les données sont organisées, et comment SQL*Loader va les insérer dans les tables.

Généralités



Généralités

- Fichiers SQL*Loader
  - Fichier de contrôle (fait partie des paramètres de l'utilitaire SQL*Loader, afin de lui indiquer comment charger les données dans la base oracle)
  - Fichier de données
  - Fichier Log (rend compte de l'exécution de l'utilitaire)
  - Fichier Bad (contient les données qui n'ont pas pu être chargées)
  - Fichier Discard (contient les données qui ne correspondent pas aux critère énoncés dans le fichier de contrôle)

Les paramètres de SQL*Loader

Parameter	Description
`userid`	The database connect string, for example, `scott/tiger@prod`).
`control`	The name of the control file.
`log`	The name of the log file. The default is the control filename with a `.log` extension.
`bad`	The name of the bad file. The default is the datafile name, but with a `.bad` extension.
`data`	The name of the datafile. The default is the control filename with a `.dat` extension.
`discard`	The name of the discard file. The default is the datafile name, but with a `.dsc` extension.
`discardmax`	The maximum number of discards to allow before failing. The default is all.
`skip`	The number of records to skip before starting to load. The default is none.
`load`	The number of records to load. The default is all.

Les paramètres de SQL*Loader

`errors`	The number of errors to allow before failing. The default is 50.
`rows`	The number of rows in a conventional path bind array or between direct path data saves. The default is 64 rows in conventional path mode and all rows in direct path mode.
`bindsize`	The size of the conventional path bind array in bytes. The default is 256KB.
`direct`	If `TRUE`, use direct path. The default is `FALSE`, indicating conventional path.
`parfile`	The name of a file containing additional command-line parameters. There is no default.

Fichier de contrôle

- Les données à charger peuvent être placées dans le fichier contrôle ou dans un fichier séparé contenant les données.
- Le fichier de contrôle fournit les informations suivantes à oracle:
  - Nom des fichiers de données
  - Types de données des champs de ces fichiers
  - Comment chaque champ est Délimité
  - Quelles sont les tables et colonnes à charger
  - ….

Fichier de contrôle

- Le fichier de contrôle
  - Décrire les données à charger
  - Indiquer les tables de la base à charger
  - Décrire l'inter-dépendance entre les données et les colonnes des tables
- Indications pour le fichier de contrôle
  - Ecrire le fichier de contrôle en format libre en minuscules ou majuscules (casse).
  - La casse n'est pas importante sauf pour les chaînes de caractères entre quote.
  - Faire précéder les commentaires de 2 tirets
  - Les mots réservés SQL*Loader utilisés pour les noms de tables ou de colonnes doivent être entre quote.

Fichier de contrôle: Le chargement de données de longueur variable

Dans ce cas, il est nécessaire que les champs de ces données soient délimités

```
sqlldr hr/hr control=regions.ctl
```

The control file `regions.ctl` contains the following:

```
LOAD DATA
-- Control file begins with LOAD DATA
INFILE *
-- The * tells SQL*Loader the data is inline
INTO TABLE regions TRUNCATE
-- truncate the target table before loading
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
-- how to parse the data
 (region_id, region_name)
-- positional mapping of data file fields to table columns
-- lines following BEGINDATA are loaded
-- no comments are allowed after BEGINDATA
BEGINDATA
1,"Europe"
2,"Americas"
3,"Asia"
4."Middle East and Africa"
```

Fichier de contrôle: Le chargement de données de longueur fixe

Dans ce cas, il est nécessaire de donner la position et la longueur de chaque champs dans le fichier de contrôle. Il est nécessaire de spécifier le type de données du champ

alone file called `regions.dat` and is in the following pipe-delimited, fixed format:

```
1|Europe                |
2|Americas               |
3|Asia                   |
4|Middle East and Africa |
```

The command line is:

```
sqlldr hr/hr control=regions.ctl
```

The content of the control file is:

```
LOAD DATA
INFILE  '/apps/seed_data/regions.dat'
BADFILE '/apps/seed_data/regions.bad'
DISCARDFILE '/apps/seed_data/regions.dsc'
OPTIONS (DIRECT=TRUE)
-- data file spec
INTO TABLE regions APPEND
-- add this data to the existing target table
(region_id   POSITION(1)    INTEGER EXTERNAL
,region_name POSITION(3:25) NULLIF region_name = BLANKS
) -- how to parse the data
```

Fichier de contrôle

- Fichier de paramètre pour le chargement

TstPara.par

USERID=hr/hr

CONTROL=c:\TstCont.ctl

Direct=true

Rows=256

c:\>sqlldr parfile=d:\TstPara.par

Fichier Log

- Contenu du fichier log
  - Les erreurs trouvées pendant la phase d'analyse du fichier contrôle sont stockées dans le fichier log.
  - Des informations détaillées sur le chargement sont stockées dans le fichier log.

Fichier Log

```
SQL*Loader: Release 10.1.0.2.0 - Production on Mar. DÚc. 22 00:13:32 2009

Copyright (c) 1982, 2004, Oracle.  All rights reserved.
Control File: $1$DUA12:[TRAIN.NF7.TEACH_1.DO]PARTSDATA3.CTL
Data File:    $1$DUA12:[TRAIN.NF7.TEACH_1.DO]PARTSDATA3.DAT
Bad File:     $1$DUA12:[TRAIN.NF7.TEACH_1.DO]PARTSDATA3.BAD
Discard File: none specified
(Allow all discards)

Number to load:  ALL
Number to skip:  0
Errors allowed:  50
Bind array:      64 row, maximum of 65536 bytes
Continuation:    Concatenate every 2 physical records
Path used:       Conventional
```

Fichier Log

```
Table PARTS, loaded from every logical record.
Insert option in effect for this table: REPLACE
Column Name  Position   Len  Term  Encl  Data Type
-----------  --------  ---  ----  ----  ----------
PARTNO          FIRST     *      ,  O(")  CHARACTER
DESCX            NEXT     *      ,  O(")  CHARACTER
QTY              NEXT     *      ,  O(")  NUMBER
Record 10: Rejected - Error on table PARTS, column QTY.
ORA-01722: invalid number
           .........
Record 70: Rejected - Error on table PARTS, column QTY.
ORA-01722: invalid number
Table PARTS: 76 Rows successfully loaded.
4 Rows not loaded due to data errors.
0 Rows not loaded because all WHEN clauses were failed.
```

Note: ceci n'est pas la totatilité du fichier log

Fichier Log

```
Record 70: Rejected - Error on table PARTS, column QTY.
ORA-01722: invalid number
Table PARTS: 76 Rows successfully loaded.
         4 Rows not loaded due to data errors.
         0 Rows not loaded because all WHEN clauses were failed.
         0 Rows not loaded because all fields were null.
Space allocated for bind array:   49920 bytes(64 rows)
Space allocated for memory besides bind array:   76956 bytes
Total logical records skipped: 0
Total logical records read: 80
Total logical records rejected: 4
Total logical records discarded:  0
Run began on Thu Aug 17 08:04:24 1995
Run ended on Thu Aug 17 08:04:28 1995
Elapsed time was:   00:00:04.16
CPU time was:       00:00:02.22
```

Fichier Bad

- Causes de rejets des enregistrements
  - Format d'entrée invalide
  - Ligne invalide
  - Valeur de clé non unique
  - Un champ Null est demandé

Stockage des Enregistrements Ecartés

- Le fichier Discard
  - Contient tous les détails des enregistrements écartés (éliminés), car ils ne coïncident pas aux critères indiqués dans la clause WHEN du fichier de contrôle.
  - Peut contenir des enregistrements jusqu'au maximum spécifié, après quoi le chargement sera interrompu.

521

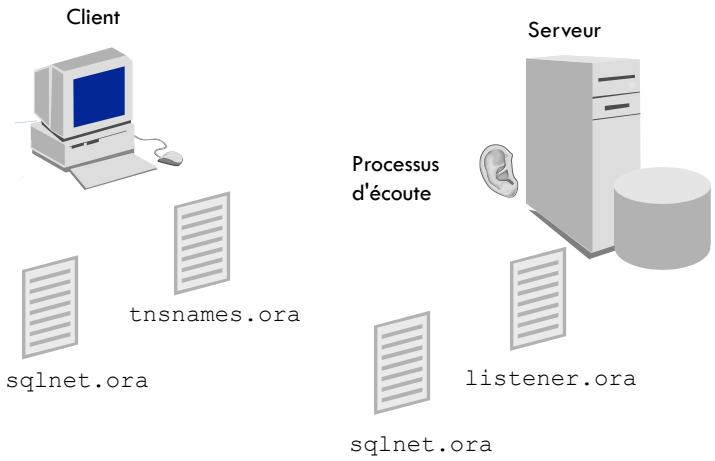
Oracle Net (Configuration du réseau)

Oracle Net

- Une fois votre instance de base de données crée, paramétrée et démarrée, elle est prête à l'utilisation, mais peut être voudrez vous (c'est généralement le cas) pouvoir y accéder via le réseau.

- Oracle Net est la solution réseau d'oracle qui permet l'utilisation de l'architecture client/serveur dans les applications de base de données.
  - Le cœur d'oracle Net est le listener (processus d'écoute)

Oracle Net



Oracle Net

524

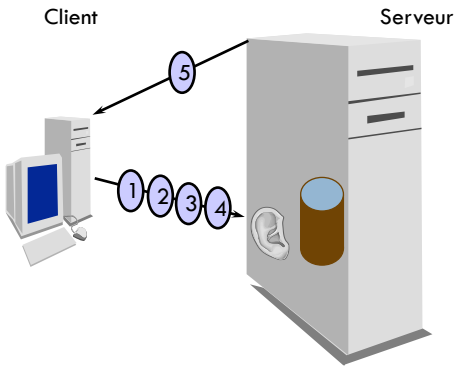
- Un Listener est un processus serveur chargé de détecter les requêtes de connexion à un service.
  - Lorsqu'un client émet une requête de connexion à une base de données oracle, le Listener actif sur le serveur intercepte la demande et fait le nécessaire pour mettre en relation le processus utilisateur avec le processus serveur de l'instance qui est associée à la BDD sollicitée.

Oracle Net

525

1. Un client cherche à se connecter à un nom de service

   connect system/manager@biblio (exemple biblio)
1. Le nom de service est résolu par une certaine méthode en un descripteur de connexion comportant l'adresse du service. Son nom et le protocole à utiliser
2. La demande de connexion est ensuite envoyée à l'adresse mentionnée
3. À l'adresse mentionnée, un listener doit être présent pour recevoir la demande et la transmettre au service…
4. …qui établit la connexion



Oracle Net

526

- Quand la connexion est allouée, le processus Listener, suivant l'architecture de la BDD va affecter le processus client à un processus serveur (architecture dédié) ou un processus dispacher (partagé)

Coté serveur (`listener.ora`)

527

- Les configurations sont enregistrées dans le fichier listener.ora
  - `%ORACLE_HOME%\network\admin`
- Deux façons de procéder
  - à la main, en éditant le fichier listener.ora
  - avec les assistants
    - Net Configuration assistant

Coté serveur (listener.ora)

Lorsque le logiciel Oracle est installé, le fichier `listener.ora` est créé pour la première base de données avec les paramètres par défaut suivants :

- Nom du processus d'écoute — LISTENER
- Port — 1521
- Protocoles — TCP/IP et IPC
- Nom SID — Instance par défaut
- Nom d'hôte — Nom d'hôte par défaut

Coté serveur (listener.ora)

```
LISTENER =
(ADDRESS_LIST =
 (ADDRESS= (PROTOCOL= TCP)(Host= stc-
     sun02)(Port= 1521)))
SID_LIST_LISTENER =
 (SID_LIST =
  (SID_DESC =
    (ORACLE_HOME= /home/oracle)
           (SID_NAME = BIBLIOTHEQUE)))
```

Utilitaire de contrôle du processus d'écoute (LSNRCTL)

Les commandes de l'utilitaire de contrôle du processus d'écoute peuvent être exécutées à partir de la ligne de commande ou de l'invite LSNRCTL.

- Syntaxe de ligne de commande UNIX :

```
$ lsnrctl <command name>
```

- Syntaxe de l'invite :

```
LSNRCTL> <command name>
```

Commandes de LSNRCTL

Les commandes suivantes permettent de gérer le processus d'écoute :

- `START [listener_name]`
- `STOP  [listener_name]`

Coté Client (le tnsnames.ora & le sqlnet.ora)

- Les fichiers (le tnsnames.ora & le sqlnet.ora) utiles à la configuration d'un client SQL*Net se trouvent :
  - Son emplacement par défaut est $ORACLE_HOME/network/admin sous UNIX et %ORACLE_HOME%\network\admin sous Windows NT.
- Le client doit pouvoir traduire les alias SQL*Net qui lui sont fournis dans la chaine de connexion.
  - Il existe essentiellement trois méthodes de résolution de nom de base distante.

Coté Client (le tnsnames.ora & le sqlnet.ora)

- un fichier **sqlnet.ora** précise s'il existe l'ordre (les priorités) des méthodes de résolution de nom utilisées par le client à l'aide du paramètre NAMES.DIRECTORY_PATH
- La résolution de noms peut être
  - locale via le fichier tnsnames.ora,
  - centralisée par un serveur de nom spécifique le serveur OracleNames,
  - fournie directement par la DNS : méthode HOSTNAME (en SQL*Net TCP/IP uniquement bien sûr...)
- par défaut (par exemple s'il n'existe pas de fichier SQLNET.ORA) : dans l'ordre TNSNAMES, ONAMES, HOSTNAME

Fichiers générés : sqlnet.ora

```
# SQLNET.ORA Network Configuration File:
/u03/ora9i/rel12/network/admin/sqlnet.ora
# Generated by Oracle configuration tools.
NAMES.DIRECTORY_PATH= (TNSNAMES, HOSTNAME)
SQLNET.EXPIRE_TIME=0
```

```
sqlplus system/manager@MY_SERVICE
SQL*Plus:Release 9.0.1.0.0-Production on Thu Nov 15 13:46:24 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
Connected to:
Oracle9i Enterprise Edition Release 9.0.1.0.0 - Production
JServer Release 9.0.1.0.0 - Production
SQL>
```

Résolution locale via tnsnames.ora

- Il doit préciser essentiellement le protocole utilisé, le nom ou l'adresse IP de la machine cible, le cas échéant (TCP/IP) : le port d'écoute du serveur (1521 ou 1525 par défaut) et l'identifiant de la base sur le serveur (ORACLE_SID).

```
# TNSNAMES.ORA Network Configuration
# File:/u03/ora9i/rel12/network/admin/tnsnames.ora
# Generated by Oracle configuration tools.
BIBLIO =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS =
     (PROTOCOL = TCP)(HOST = stc-sun02)(PORT =1521))
   )
   (CONNECT_DATA =
     (SERVICE_NAME = BIBLIOTHEQUE)
   )
 )
```