



Oracle 11g

Administration

Auteur : Clotilde Attouche

Société TELLORA

Version 1.2

Du 6 Mai 2010



Sommaire

1	Présentation	10
1.1	Les produit Database proposes par Oracle.....	11
1.2	Notion de Grid Computing.....	12
1.2.1	La gestion ASM (gestionnaire de fichiers : Automatic Storage Management)	13
1.2.2	Les composants développés par Oracle pour le Grid Computing	14
1.2.3	Outils de développement	14
1.3	Règles de nommage dans Oracle Database	14
2	La documentation	15
2.1	Le support oracle	15
3	Notion de schéma	17
4	Le dictionnaire de données	18
4.1	Les tables et vues statiques	18
4.2	Les tables et vues dynamiques de performance	19
5	Outils d'administration	20
5.1	L'outil SQL*Plus.....	20
5.1.1	Environnement de travail.....	21
5.1.2	Quelques commandes SQL*Plus	21
5.2	L'outil iSQL*Plus	22
5.3	Le Database Control et le Grid control	23
6	L'architecture OFA	26
7	Installation Oracle	28
7.1	Pré-requis matériel	28
7.2	Installation du client.....	30
8	Architecture Oracle	31
8.1	Connexion utilisateur.....	33
8.1.1	La PGA (Program Global Area).....	33
8.1.2	La SGA: System Global Area.....	33
8.2	Le fichier de paramètres (init.ora ou SPFILE.ORA).....	35
8.3	Les processus d'arrière plan	36
8.4	La base de données	37
8.4.1	Les fichiers de données.....	37
9	Utilisateurs et connexion à la base de données.....	39
9.1	Syntaxe pour la connexion classique	39
9.2	Syntaxe pour la connexion spéciale SYSDBA ou SYSOPER	39
9.3	Les connexions SYSDBA et SYSOPER.....	40
9.4	Le fichier de mots de passe.....	40
9.5	Les variables d'environnement	41
10	Démarrer & Arrêter une base de données	42
10.1	Démarrer la base de données	43
10.2	Modifier la disponibilité de la base de données.....	44
10.3	Arrêter la base de données	45
10.4	Ouvrir la base de données en mode RESTRICT	46



10.5	Mettre l'instance dans un état QUIESCE.....	47
10.6	Vues du dictionnaire de données.....	47
11	Gestion de l'instance et SPFILE	49
11.1	Créer le fichier du paramètre SPFILE	49
11.2	Exporter un fichier de paramètres serveur SPFILE.....	50
11.3	Modifier des paramètres de l'instance ou du SPFILE	51
11.4	Vues du dictionnaire de données.....	52
12	Créer une base de données	53
12.1	Présentation du script de création de la base.....	54
12.2	Présentation de l'outil DBCA	57
12.3	Valeurs des paramètres	60
12.4	Vues du dictionnaire de données.....	63
12.5	EMCA : Création de l'OEM repository (Database Control)	63
13	Automatiser le démarrage de la base.....	66
13.1	Sous unix	66
13.2	Sous Windows	66
14	Accéder à une base distante.....	68
14.1	Configuration coté serveur	69
14.2	Configuration coté client	71
14.3	Changer de machine automatiquement.....	72
14.4	EZCONNECT.....	72
14.5	Bases distantes et database Links	73
15	Sécuriser la base de données	75
15.1	Le fichier de contrôle.....	75
15.2	Protection du fichier de contrôle.....	77
15.2.1	Multiplexer le fichier de contrôle	78
15.3	Vues du dictionnaire de données.....	79
15.4	Protection des fichiers de Redo Log	79
15.4.1	Dimensionner les fichiers de Redo Log	80
15.4.2	Multiplexer les fichiers de Redo Log	81
15.4.3	Ajouter un groupe de Redo Log	82
15.4.4	Déplacer les fichiers de Redo Log	82
15.4.5	Supprimer un groupe de fichiers redo log	83
15.4.6	Supprimer un membre d'un groupe de redo log.....	83
15.4.7	Forcer le basculement du groupe courant.....	84
15.4.8	Trouver des informations sur les fichiers Redo Log	85
16	Gestion du stockage.....	87
16.1	Notion de tablespace	87
16.2	Organisation du stockage dans un tablespace.....	89
16.3	Notion de BIGFILE ou de SMALLFILE	91
17	Tablespaces permanents.....	93
17.1	Créer un tablespace permanent	93
17.2	Modifier un tablespace permanent.....	94
17.2.1	Agrandir un tablespace	94
17.2.2	Passer un tablespace OFFLINE ou ONLINE :	95
17.2.3	Passer un tablespace READ ONLY ou READ WRITE :	95
17.2.4	Déplacer un fichier de données.....	96



17.2.5	Renommer un tablespace	97
17.2.6	Supprimer un tablespace	99
17.2.7	Créer un tablespace avec une taille de bloc non standard.....	100
17.2.8	Le cryptage des données d'un tablespace (nouveau 11g).....	100
17.2.9	Tablespace de travail par défaut.....	101
17.2.10	Vues du dictionnaire de données	102
18	Tablespaces SYSTEM & SYSAUX	103
18.1	Tablespace SYSTEM	103
18.2	Tablespace SYSAUX	103
18.2.1	Avantages du tablespace SYSAUX.....	104
18.2.2	Délocaliser les occupants du tablespace SYSAUX.....	104
19	Tablespace UNDO	106
19.1	Fonctionnement du tablespace UNDO	108
19.2	Positionner les paramètres de gestion automatique.....	109
19.3	Créer un tablespace UNDO	109
19.4	Changer de tablespace UNDO actif.....	110
19.4.1	Créer un tablespace UNDO après création de la base.....	110
19.4.2	Changer de tablespace UNDO pendant l'activité de la base.....	111
19.5	Administrer un tablespace UNDO	112
19.5.1	Dimensionner le tablespace UNDO.....	113
19.5.2	Supprimer un tablespace UNDO.....	114
19.6	Vues du dictionnaire de données.....	114
20	Tablespaces temporaires.....	117
20.1	Créer un tablespace temporaire	119
20.2	Groupes de tablespaces temporaires	119
20.3	Administrer les tablespaces temporaires	120
20.3.1	Agrandir un tablespace temporaire.....	121
20.3.2	Modifier la clause AUTOEXTEND :.....	121
20.3.3	Modifier la taille d'un fichier temporaire	122
20.3.4	Rétrécir un tablespace temporaire (Nouveautés 11g)	122
20.3.5	Supprimer un tablespace temporaire	123
20.4	Définir un tablespace temporaire par défaut.....	123
20.5	Vues du dictionnaire de données.....	124
21	Monitoring de l'utilisation d'un tablespace	125
21.1	Configuration des seuils de tablespace.....	126
22	Mémoire dynamique et performances	128
22.1	La notion de granule	128
22.2	Gestion automatique du partage de la mémoire en 10g	128
22.2.1	Principes de tuning de la SGA	129
22.2.2	SGA_TARGET et le Database Control (OEM)	130
22.2.3	Configuration manuelle de SGA_TARGET	131
22.2.4	Comportement des paramètres Auto-tuned	131
22.2.5	Comportement des paramètres Manuels.....	132
22.2.6	Redimensionner SGA_TARGET	133
22.2.7	Désactiver la gestion automatique de la mémoire en version 10g	133
22.3	Gestion automatique de la mémoire en 11g	133
22.3.1	Désactiver la gestion automatique de la mémoire en version 11g	134
22.3.2	La vue dynamique V\$MEMORY_TARGET_ADVICE.....	134
22.3.3	Nouveau cache en version 11g : result_cache	134



22.4	L'optimiseur Oracle	134
22.4.1	Les optimiseurs RBO et CBO	135
22.4.2	Présentation du SQL Tuning Advisor	136
22.4.3	Impacte sur les Statistiques	137
22.5	L'optimiseur Oracle et la gestion des statistiques	138
22.5.1	Statistiques sur les tables.....	138
22.5.2	Interpréter les statistiques générées sur les tables.....	140
22.5.3	Statistiques sur les index.....	141
22.5.4	Problèmes détectés sur les index.....	142
22.6	Outil de collecte des statistiques	142
22.6.1	GATHER_STATS_JOB.....	143
22.6.2	Modifier l'exécution des statistiques.....	144
22.7	Automatic Database Diagnostic Monitor (ADDM)	144
22.7.1	Méthode d'analyse utilisée par ADDM	146
22.7.2	Résultats de l'analyse ADDM dans le grid Control	147
22.7.3	Recommandations d'ADDM	148
22.7.4	Nouvelles vues en version 11g pour ADDM	148
22.7.5	Exemple de génération de rapport ADDM	149
23	La gestion des utilisateurs	154
23.1.1	Création d'un utilisateurs identifié par le système d'exploitation	154
23.1.2	Création d'un utilisateurs identifié par Oracle	156
23.1.3	Modification d'un utilisateur dans Oracle.....	157
23.1.4	Suppression d'un utilisateur	158
23.1.5	Supervision des utilisateurs.....	158
23.2	Les Profils	159
23.2.1	Activer la limitation des ressources	159
23.2.2	Création d'un profil	160
23.2.3	Modification d'un profil	161
23.2.4	Suppression d'un profil.....	162
23.2.5	Vues du dictionnaire de données	163
23.3	Le gestionnaire de ressources.....	163
23.3.1	Les groupes de consommation de ressources : Consumers Groups.....	164
23.3.2	Le package DBMS_RESSOURCE_MANAGER	165
23.3.3	Consumers groups et plan de ressources.....	165
23.3.4	Assigner des priorités	167
23.4	La gestion des droits	173
23.4.1	Privilèges systèmes	173
23.4.2	Privilèges objets	175
23.5	Les Rôles	178
23.5.1	Création d'un rôle	178
23.5.2	Attribuer des privilèges à un rôle.....	179
23.5.3	Enlever des privilèges à un rôle	179
23.5.4	Attribuer un rôle à un utilisateur ou à un autre rôle	180
23.5.5	Enlever un rôle à un utilisateur ou à un rôle	180
23.5.6	Supprimer un rôle.....	181
23.5.7	Activer ou désactiver un rôle	181
23.5.8	Définir des rôles par défaut	182
23.5.9	Rôles prédéfinis	182
23.6	Supervision des utilisateurs connectés.....	183
23.7	Déconnecter un utilisateur	183



23.8	Vues du dictionnaire de données.....	183
24	Les Objets de stockage	185
25	Les tables.....	186
25.1	Les contraintes d'intégrité.....	187
25.1.1	Contraintes immédiates ou différées	187
25.1.2	Créer des contraintes d'intégrité	188
25.1.3	Désactiver les contraintes d'intégrité	189
25.2	Les colonnes virtuelles en 11g	189
25.3	Le ROWID	190
25.4	Bloc Oracle et lignes de tables	191
25.5	La High Water Mark	192
25.6	Pilotage du remplissage d'un bloc	193
25.6.1	Calcul de PCTFREE et PCTUSED	194
25.6.2	Directives pour PCTFREE et PCTUSED	195
25.7	Spécifier le stockage d'une table.....	195
25.8	Chaînage et migrations.....	197
25.9	Rappel sur la génération des statistiques	197
25.9.1	Statistiques pour les tables dans le dictionnaire de données	198
25.9.2	Interpréter les statistiques générées.....	199
25.9.3	Le package DBMS_SPACE.....	199
25.10	Réorganiser le stockage d'une table	200
25.10.1	L'ordre SQL Alter table ...Move	200
25.10.2	L'ordre SQL : SHRINK	201
25.11	Libérer de l'espace dans un segment.....	204
25.12	Le monitoring d'une table	205
25.13	Vues du dictionnaire de données.....	206
26	Les index.....	207
26.1	Organisation logique	207
26.2	Organisation physique	208
26.3	Accès par index B*-Tree	210
26.4	Index sur fonctions	211
26.5	Index Bitmap	212
26.6	Index IOT	214
26.7	Index à clé inversée	215
26.8	Créer et Spécifier le stockage d'un index.....	217
26.8.1	La clause USING INDEX	218
26.9	Calcul de PCTFREE	220
26.10	Rappel sur l'analyse des index	221
26.10.1	Problèmes détectés	222
26.11	Réorganiser le stockage d'un index	222
26.11.1	Comment reconstruire des index dégradés	223
26.11.2	L'ordre SQL ALTER INDEX ... REBUILD	224
26.11.3	L'ordre SQL ALTER INDEX ... COALESCE	224
26.11.4	L'ordre SQL ALTER INDEX ... SHRINK SPACE.....	225
26.12	Surveiller l'utilisation des index.....	225
26.13	Supprimer un index.....	226
26.14	Vues du dictionnaire de données.....	227
27	Les partitions de tables et d'index	228



27.1.1	Performance des partitions.....	228
27.2	Vues du dictionnaire de données.....	230
28	Le Scheduler (CJQ).....	231
28.1	Concepts du scheduler.....	233
28.2	Privilèges associés au Scheduler	234
28.2.1	Privilèges utilisateurs	235
28.2.2	Privilèges administrateurs	235
28.3	Créer et gérer un programme dans un schedule	236
28.4	Les JOBS	236
28.4.1	Créer un JOB	236
28.4.2	Spécifier des schedules pour un job.....	238
28.4.3	Créer et utiliser des schedules.....	239
28.5	Les JOB CLASS	240
28.5.1	Créer une JOB CLASS	240
28.6	Gestion des Logues de JOB.....	241
28.6.1	Le package DBMS_SCHEDULER	241
28.6.2	Les logs des JOBS	242
28.7	Les Windows	243
28.7.1	Créer une WINDOW.....	243
28.7.2	Attribuer des priorités aux JOBS dans les WINDOWS.....	244
28.8	Activer un composant du scheduler	245
28.9	Gérer des composants du Scheduler.....	245
28.9.1	Gérer un JOB.....	245
28.9.2	Gérer un PROGRAM	246
28.9.3	Gérer un schedule.....	247
28.9.4	Gérer une fenêtre de temps : Window	247
28.9.5	Priorité des Windows	248
28.9.6	Gérer les attributs des composants du scheduler	249
28.10	Vues du dictionnaire de données.....	250
29	Jeux de caractères et paramètres NLS	253
29.1	Introduction	253
29.2	Migration de jeux de caractères	256
29.2.1	Migration du jeu de caractères par EXPORT/IMPORT	257
	Vues du dictionnaire de données.....	258
30	Présentation de l'utilitaire DATA Pump	259
30.1	Opérations d'IMPORT et d'EXPORT du DATA Pump	260
30.2	Avantages de l'export et de l'import DATA Pump:.....	261
30.3	Le mode interactif du DATA Pump	262
30.3.1	Commandes du mode interactif	262
30.4	Méthode d'extraction des données avant et après <i>data pump</i>	263
30.4.1	Méthode direct path (chemin direct) du <i>data pump</i>	263
30.4.2	Données conduisant un accès utilisant des tables externes :.....	264
31	Export/Import Data Pump	265
31.1	Fichiers supportés par les outils <i>DATA Pump</i>	265
31.2	Paramètres le l'export et de l'import DATA Pump	266
31.2.1	Paramètres communs	266
31.2.2	Paramètres de l'EXPORT DATA Pump.....	268
31.2.3	Paramètres de l'IMPORT DATA Pump.....	268



31.3	Filtrer les données à exporter	268
31.4	Exemples d'export et d'import DATA Pump.....	270
31.4.1	Estimation de la taille de l'Export	270
31.4.2	Exports Parallélisés	270
31.4.3	Import Parallélisé.....	271
31.4.4	Export de schéma	271
31.4.5	Import de schéma.....	272
31.5	Remarques et modes opératoires.....	272
31.5.1	Export et jeux de caractères	272
31.5.2	Remarques sur les dépendances entre les objets	273
31.5.3	Export de niveau tablespace	273
31.6	Vues du dictionnaire de données de <i>DATA Pump</i>	274
32	SQL*Loader.....	276
32.1	Fichier de paramètres.....	277
32.2	Le fichier de contrôle.....	279
32.3	Exemples de chargements	280
32.3.1	Exemples de fichiers de contrôle : Longueur variable enregistrements	281
32.3.2	Exemples de fichiers de contrôle : Longueur fixe avec élimination d'enregistrements.....	282
32.3.3	Chargement dans deux tables.....	282
32.3.4	Chargement dans deux tables avec utilisation d'une colonne FILLER	283
32.4	Chargement de données LOB	284
32.5	Chargement de formats XML	284
33	Stratégie de Sauvegardes et Restaurations.....	287
33.1	Les modes NOARCHIVELOG et ARCHIVELOG	288
33.1.1	Le mode NOARCHIVELOG.....	289
33.1.2	Le mode ARCHIVELOG	289
33.1.3	Mettre la base en mode ARCHIVELOG	290
33.1.4	Les paramètres du processus ARCH.....	290
33.2	Passer la base en mode ARCHIVELOG.....	291
33.3	Administrer le processus <i>ARCH</i>	292
33.3.1	Forcer l'archivage de façon périodique	292
34	Sauvegardes	294
34.1	Sauvegarde base arrêtée	295
34.2	Sauvegarde base en ligne	296
34.2.1	Sauvegarde du fichier de contrôle	296
34.3	Sauvegarde partielle d'un tablespace <i>ONLINE</i>	297
34.4	Sauvegarde de tous les tablespaces de la base <i>ONLINE</i>	298
34.5	Vues du dictionnaire de données.....	299
34.6	Stratégie recommandée par Oracle	300
34.7	Recover Manager (RMAN)	300
34.8	Le Flash Back	302
35	Procédures de sauvegardes	303
35.1	Sauvegardes logiques.....	303
35.2	Sauvegardes hors ligne	305
35.3	Sauvegardes en ligne	305
35.4	RMAN	307
35.5	Planification et intégration de procédures.....	308



35.6	Intégration des sauvegardes Oracle et du système	310
35.7	Les grandes bases de données	311
36	Restaurations.....	314
36.1	La commande RECOVER	314
36.1.1	Exemples de restaurations.....	315



1 Présentation

La version oracle database 11g release 2 est disponible depuis septembre 2010.

La version 11.2 pour Windows est disponible depuis avril 2010.

Cette nouvelle release contient l'outil de développement rapide *APEX (Oracle Application Express)*.

Un serveur http est également intégré dans la base de données. Il utilise la technologie WebDAV et est implémenté sous le nom de *XML DB*. Il est nommé par Oracle « *Embedded PL/SQL Gateway* ».

Oracle Database 11g représente la nouvelle génération de la gestion des informations en entreprise, qui permet de faire face aux exigences qu'imposent la croissance rapide des volumes de données, l'évolution constante de l'environnement et la nécessité de fournir une qualité de service maximale tout en réduisant et en contrôlant les coûts informatiques. Oracle 11g offre une performance améliorée du stockage sur fichiers, des fonctionnalités renforcées pour la sécurité, d'importantes améliorations de performances pour Oracle XML DB, et des fonctions nouvelles pour l'OLAP et le datawarehouse.

Oracle Database 11g reste centré sur le grid computing : il permet de constituer des matrices de serveurs et de systèmes de stockage économiques, capables de traiter les données de façon rapide, fiable et évolutive, en supportant les environnements les plus exigeants, qu'il s'agisse de datawarehouse, de transactionnel ou de gestion de contenus.

Oracle 11g multiplie les outils de gestion et introduit de nouvelles fonctionnalités d'auto gestion et d'automatisation. *Automatic SQL*, *Partitioning Advisor* ou *Support Workbench* accompagnent les administrateurs pour améliorer les performances et les informer le plus rapidement possible des incidents.

Ainsi

—*Oracle Flashback Transaction* permet de revenir plus facilement sur une erreur de transaction et de dépendances.

—*Parallel Backup and Restore* augmente les performances des sauvegardes sur les grosses bases de données.

—*Hot Patching* permet d'appliquer les mises à jour sans arrêter les bases.

—*Data Recovery Advisor* accompagne les administrateurs pour déterminer intelligemment les plans de secours.

—*Oracle Fast Files* adopte un comportement proche des systèmes de gestion de fichiers, ce qui est un gage de performances avec les objets de type LOBs (*Large Objects*) ou les fichiers contenant du texte, des images, des données XML ou encore les objets tridimensionnels.

—*Oracle XML DB* permet de stocker et manipuler nativement les données XML. Le langage XML se révèle « lourd », et avec cette approche Oracle 11g limite la dégradation de ses performances. De même la base supporte les interfaces standard *XQuery*, *Java Specification Requests (JSR)-170* et *SQL/XML*.

—*Oracle Transparent Data Encryption* permet de crypter les données des tables, des index ou encore les données stockées de type LOB.—*Cubes OLAP* apporte des fonctionnalités de datawarehouse (*fermes de données*), Oracle 11g embarque les cubes OLAP pour visualiser les informations stockées, ce qui autorise le développement de requêtes au format SQL.

—*Continuous Query Notification* notifie immédiatement les changements apportés dans la base de données.

—avec *Query Result Caches*, requêtes et fonctionnalité de la base ou d'applications tierces sont placées en cache afin d'optimiser leur accès.



—*Database Resident Connection Pooling* est destiné aux applications qui ne sont pas *multithreadées* (en *multithreaded server* : MTS), par exemple pour certains systèmes Web, Oracle 11g permet de créer des « *pools* » de connexions en *multithreaded server* (MTS).

1.1 Les produit Database proposes par Oracle

Les différents produits d'Oracle DATABASE sont proposés en trois gammes

- ♦ **Enterprise Edition** - La gamme pour les grosses applications critiques de l'entreprise, intégrant des options supplémentaires telles que le partitionnement des tables.
- ♦ **Standard Edition** - La gamme destinée à des serveurs possédant 4 processeurs et ne proposant que l'option RAC/ASM.
- ♦ **Standard Edition ONE** - la gamme destinée aux serveurs biprocesseurs, sans option.
- ♦ **Personal Edition** - La gamme pour l'utilisateur indépendant (développeur, consultant, ...), elle utilise un noyau Enterprise Edition.

Quatre nouvelles options apparaissent dans Oracle Database 11g Enterprise Edition

- ⇒ · Oracle Real Application Testing
- ⇒ · Oracle Advanced Compression
- ⇒ · Oracle Total Recall
- ⇒ · Oracle Active Data Guard

Oracle Real Application Testing aide ses clients à réduire les délais, les risques et les coûts de test de ses modifications de leur environnement informatique, de façon contrôlée et économique. Outil de tests et de gestion des changements, cet outil est bienvenu là où les infrastructures et environnements sont plus que jamais multiples.

Oracle Advanced Compression intègre de nouveaux mécanismes de compression applicables à tous les types de données permettant d'atteindre des taux de compression de 2x ou 3x, et parfois plus. Associé à de nouveaux mécanismes de partitionnement, Oracle Advanced Compression permet de déployer dans la base de données des stratégies de gestion du cycle de vie des informations, sans avoir à modifier les applications, afin de réduire encore plus les besoins de stockage.

Oracle Total Recall permet de conserver et de retrouver les historiques des données modifiées, mais aussi d'en simplifier l'accès. Les administrateurs peuvent intervenir plus tôt dans les processus, ce qui apporte une nouvelle dimension de temps dans la gestion des données, comme le tracking (*suivi, en temps réel des flux d'informations*), les audits ou le respect des règles.

Oracle active DATA GUARD porte la protection des données jusqu'aux risques de défaillances des systèmes et de désastres. L'application permet simultanément d'écrire et récupérer les données d'une base de données, ce qui augmente les performances et apporte une solution économique de '*Disaster Recovery*'. **Oracle Active Data Guard** peut être employé pour améliorer la performance des bases de données de production en transférant vers une base de données physique secondaire des opérations requérant beaucoup de ressources, telles que certaines requêtes ou les sauvegardes. Cette solution améliore fortement le retour sur investissement pour une base de données physique de secours, car celle-ci peut être utilisée à la fois pour la protection en cas de panne générale et pour l'amélioration de la qualité de service de l'environnement de production.



1.2 Notion de Grid Computing

A partir de la version 10g, la base de données intègre la notion de *Grid Computing* (réseau distribué d'ordinateurs hétérogènes en grille).

Le but du Grid est de créer des pools de ressources :

- ⇒ de stockage
- ⇒ de serveurs

Le Grid Computing autorise un accès transparent et évolutif (en termes de capacité de traitement et de stockage), à un réseau distribué d'ordinateurs hétérogènes.



Oracle 11g permet à ces machines d'interopérer ; l'ensemble étant considéré comme une seule ressource unifiée.
- Chaque ressource est vue comme un service !

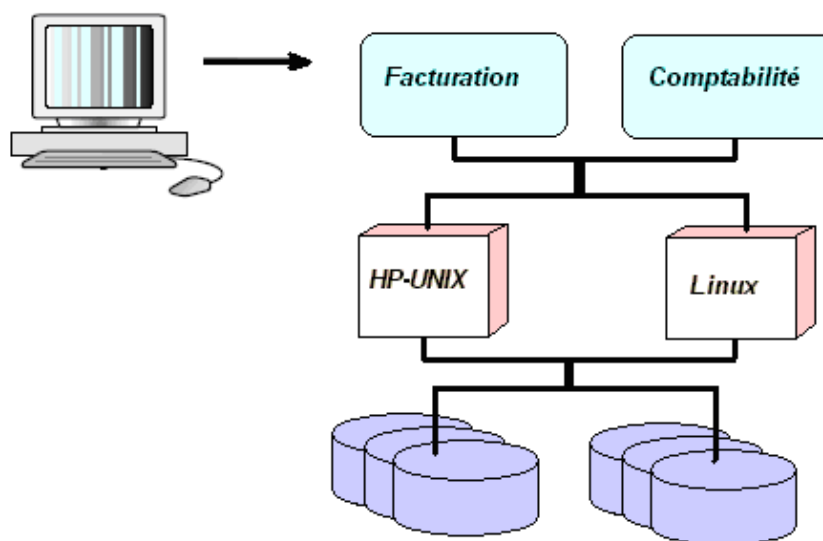
Il est possible de mettre en place des réseaux grille nationaux, voire mondiaux.

Ainsi chaque nouveau système peut être rapidement mis à disposition à partir du pool de composants

Exemple d'application en Grid Computing

Les deux applications présentées ci-dessous, Facturation et Comptabilité se partagent des ressources de deux serveurs.

- ⇒ Chacune peut être hébergée sur n'importe lequel d'entre eux et les fichiers de base de données peuvent se trouver sur n'importe quel disque.



Informatique en Grille



1.2.1 La gestion ASM (gestionnaire de fichiers : Automatic Storage Management)

La nouvelle fonctionnalité *Automatic Storage Management (ASM)* permet à la base de données de gérer directement les disques bruts, elle élimine le besoin pour un gestionnaire de fichiers de gérer à la fois des fichiers de données et des fichiers de journaux.

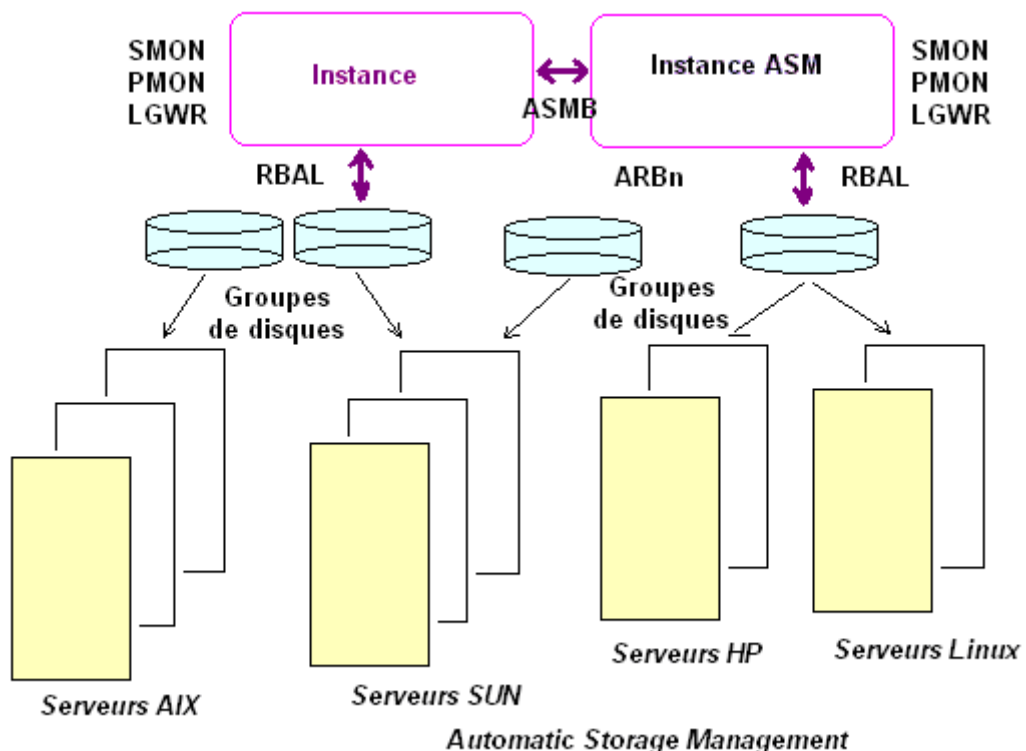
L'ASM répartit automatiquement toutes les données de bases de données entre tous les disques, délivrant le débit le plus élevé sans aucun coût de gestion.

Au fur et à mesure de l'ajout et de l'abandon de disques, l'ASM actualise automatiquement la répartition des données.

Pour utiliser ASM vous devez démarrer une instance appelée « *ASM instance* » qui doit être démarrée avant de démarrer l'instance de votre propre base de données.

Les instances ASM ne montent pas de base de données (ensemble de fichiers constituant la base) mais gère les *metadata*s requises pour rendre les fichiers ASM disponibles à n'importe quelle instance de base de données.

Les deux, instance ASM et instance « *ordinaire* » ont accès au contenu des fichiers. Communiquant avec l'instance ASM seulement pour connaître le *layout* des fichiers utilisés.



1.2.2 Les composants développés par Oracle pour le Grid Computing

- ♦ **Real Application cluster (RAC)** : Supporte l'exécution d'Oracle sur un cluster d'ordinateurs qui utilisent un logiciel de cluster indépendant de la plate forme assurant la transparence de l'interconnexion.
- ♦ **Automatic Storage Management (ASM)** : Regroupe des disques de fabricants différents dans des groupes disponibles pour toute la grille. ASM simplifie l'administration car au lieu de devoir gérer de nombreux fichiers de bases de données, on ne gère que quelques groupes de disques.
- ♦ **Oracle Ressource Manager** : Permet de contrôler l'allocation des ressources des nœuds de la grille
- ♦ **Oracle Scheduler** : Contrôle la distribution des jobs aux nœuds de la grille qui disposent de ressources non utilisées.
- ♦ **Oracle Streams** : Transfère des données entre les nœuds de la grille tout en assurant la synchronisation des copies. Représente la meilleure méthode de réplication.

1.2.3 Outils de développement

Oracle offre l'accès à un choix d'outils et processus de développement, avec de nouvelles fonctionnalités comme **Client Side Caching**, **Binary XML**, un nouveau compilateur Java, l'intégration native avec *Microsoft Visual Studio 2005* pour les applications « .NET », **Oracle Application Express** pour les outils de migration, ou encore **SQL Developer** pour coder rapidement les routines SQL et PL/SQL.

1.3 Règles de nommage dans Oracle Database

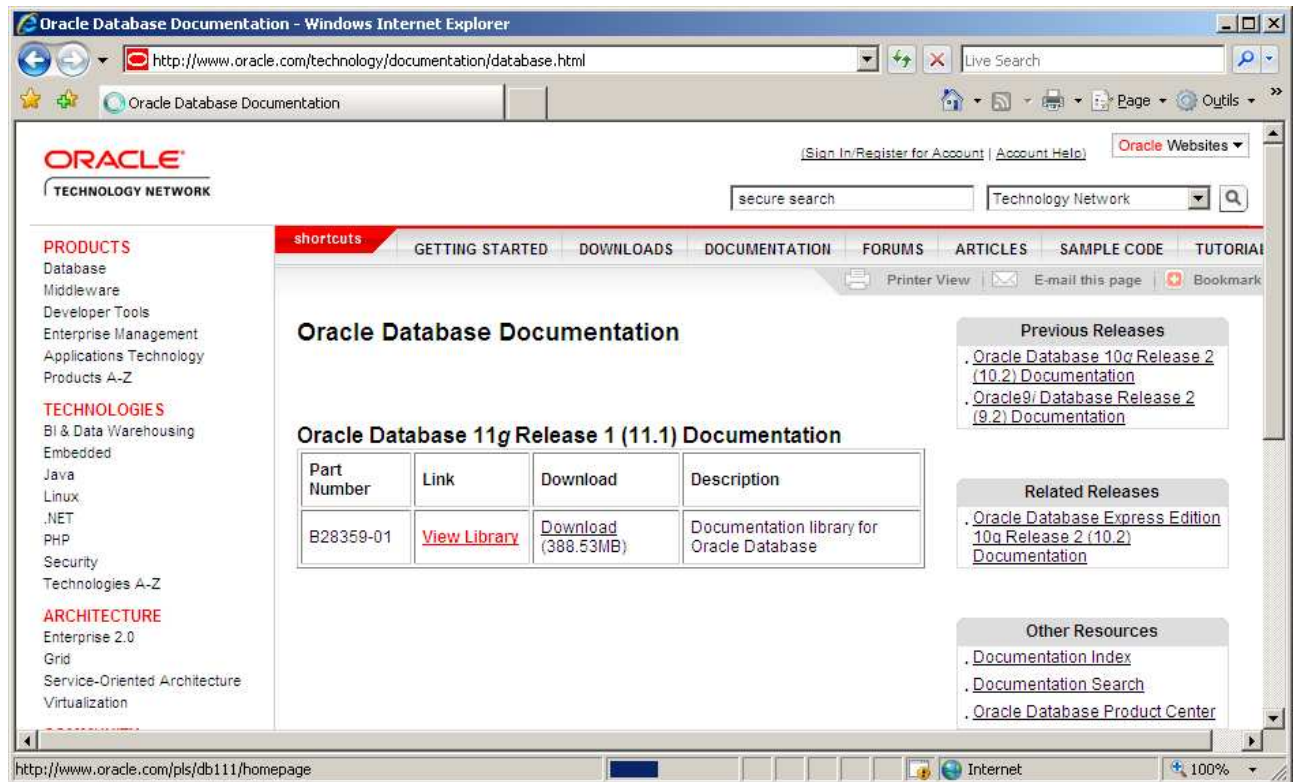
Un nom de structure Oracle doit respecter les règles suivantes :

- ♦ 30 caractères maximums
- ♦ Doit commencer par une lettre
- ♦ Peut contenir des lettres, des chiffres et certains caractères spéciaux (_\$#)
- ♦ N'est pas sensible à la casse
- ♦ Ne doit pas être un mot réservé Oracle



2 La documentation

La documentation Oracle est consultable à partir du serveur : <http://www.oracle.com>



Elle est également consultable à partir du serveur : <http://tahiti.oracle.com>

2.1 Le support oracle

Le site Metalink est le site de hotline en ligne : : <http://metalink.oracle.com> remplacé par support.oracle.com

On y trouve des résolutions d'erreurs référencées, des patches et des scripts d'administration.



The screenshot shows the My Oracle Support website in a Mozilla Firefox browser window. The address bar displays <https://support.oracle.com/CSP/ui/flash.html>. The page features the Oracle logo and the text "MY ORACLE SUPPORT". On the left, there is a section titled "My Oracle Support" with a description of the platform and a link to "Introducing Priority Handling". In the center, there is a "Sign In" section with a language dropdown set to "English" and a "Sign In..." button. Below this is a "Register here" section with links for "New user? Register here", "Read the Registration FAQ", "Troubleshooting Login Issues", and "Post-Migration FAQ". On the right, there is a "COLLECT ANALYZE ACT" banner with a list of steps: "Running JAWS? Read about installing these scripts", "Having trouble with Flash? Try these tips:", "Running Flash in your organization", "Installing the right version of Flash", "Flash FAQ", and "Adobe Flash support site". At the bottom, there is a "Tell us about your experience with My Oracle Support." link. The browser's taskbar at the bottom shows several open applications, including "Zimbra: Réception - Mozil...", "My Oracle Support - ...", "Ora11g_Administration.d...", and "Support Oracle admin 91 ...".

The screenshot shows the My Oracle Support Dashboard. The top navigation bar includes links for "Fichier", "Édition", "Affichage", "Aller à", "Signets", "Outils", "Onglets", and "Aide". The address bar shows <https://support.oracle.com/CSP/ui/flash.html#>. The dashboard header includes the Oracle logo, "MY ORACLE SUPPORT", and a "PowerView is OFF" indicator. The main content area is divided into several sections. On the left, there is a "Health Recommendations" section with a "Learn more about Health Recommendations" link. Below this is a "Getting Started" section with links for "Welcome to My Oracle Support!", "Start using Configuration Management", "My Oracle Support Training Central", and "My Oracle Support". In the center, there is a "Task: Pending User Requests" section with a "Service Requests" subsection. This section includes filters for "Technical SRs Only" and a "Contact" dropdown menu. Below the filters is a table with columns for "Problem Summary", "SR Number", "Sev...", "Contact", "Status", and "Last Upda...". The table currently shows "No information returned". On the right, there is a "Systems" section with a description: "The Systems region helps users to organize and monitor their systems. Was it working before and now it's not? Pinpoint the problem by comparing systems with history...". The dashboard also features a "Watch a video tutorial" link and a "Download" button for "IBM: Linux on POWER Sys".



3 Notion de schéma

Le terme **SCHÉMA** désigne l'ensemble des objets qui appartiennent à un utilisateur, ces objets sont préfixés par le nom de l'utilisateur qui les a créés. Il s'agit d'une notion logique désignant la totalité des objets créés par un utilisateur.

C'est ainsi que la base Oracle peut faire la différence entre la table AVION appartenant à l'utilisateur BETTY (BETTY.AVION) et la table avion appartenant à l'utilisateur CHARLY (CHARLY.AVION).



Chacun des utilisateurs propriétaire des objets à tous les droits sur ces objets !

En général on indique sous le terme de schéma, l'ensemble des tables et des index d'une même application.

Les schémas d'exemple fournis par Oracle sont décrits dans la documentation *Oracle Database Sample Schémas*.

Ces schémas peuvent être installés lors de la création de la base de données (appelée par défaut ORCL) au moment de l'installation des binaires d'Oracle.

Principaux types d'objets de schéma :

- ◆ Tables et index
- ◆ Directory
- ◆ Vues, séquences et synonymes
- ◆ Programmes PL/SQL (*procédures, fonctions, packages, triggers*)



4 Le dictionnaire de données

C'est un ensemble de tables et de vues qui donne des informations sur le contenu d'une base de données.

Il contient :

- ◆ Les structures de stockage
- ◆ Les utilisateurs et leurs droits
- ◆ Les objets (tables, vues, index, procédures, fonctions, ...)
- ◆ ...



Il appartient à l'utilisateur `SYS` et est stocké dans le tablespace `SYSTEM`.
Sauf exception, toutes les informations sont stockées en `MAJUSCULE`.

Il est créé lors de la création de la base de données, et mis à jour par Oracle lorsque des ordres `DDL` (*Data Définition Language*) sont exécutés, par exemple `CREATE`, `ALTER`, `DROP` ...

Le dictionnaire de données chargé en mémoire est utilisé par Oracle pour traiter les commandes `SQL`.

4.1 Les tables et vues statiques

Les vues statiques sont basées sur de vraies tables stockées dans le tablespace `SYSTEM`, et sont accessibles uniquement quand la base est ouverte.

Les **vues statiques** sont caractérisées par leur préfixe :

- ◆ `USER_*` : Informations sur les objets qui appartiennent à l'utilisateur
- ◆ `ALL_*` : Information sur les objets auxquels l'utilisateur a accès (les siens et ceux sur lesquels il a reçu des droits)
- ◆ `DBA_*` : Information sur tous les objets de la base

Derrière le préfixe, le reste du nom de la vue est représentatif de l'information accessible, au pluriel.



4.2 Les tables et vues dynamiques de performance

Ces tables sont basées sur des informations en mémoire ou extraites du fichier de contrôle.

Elles donnent des informations sur le fonctionnement de la base de données, notamment sur les performances. Elles sont remises à zéro si on arrête la base de données.

Elles sont accessibles même lorsque la base n'est pas complètement ouverte (MOUNT)

Les **vues dynamiques** de performance sont :

Préfixées par « V\$ »

Derrière le préfixe, le reste du nom de la vue est représentatif de l'information accessible

```
V$INSTANCE  
V$DATABASE  
V$SGA  
V$DATABASE  
V$PARAMETER
```



Les vues `DICTIONARY` et `DICT_COLUMNS` donnent la description de toutes les tables et vues du dictionnaire (statiques et dynamiques).

- la liste complète des vues statiques est obtenue par la requête :

```
SELECT view_name FROM ALL_VIEWS  
WHERE ALL_VIEWS like 'DBA*_%' escape '*'  
;
```



5 Outils d'administration

Trois outils sont présents pour administrer une base de données Oracle

- ⇒ SQL*Plus (sqlplus), interface d'accès à la base de données en mode commande
- ⇒ iSQL*Plus, peut être utilisé en application indépendante ou connecté à un référentiel Oracle *Management Server* (OMS)
- ⇒ Oracle Enterprise Manager (OEM), appelé Grid Control ou Database Control.
 - *Database control* est créé à la création d'une base oracle et ne permet d'administrer graphiquement que cette base de données.
 - *Grid control* est un outil qui permet d'administrer une ferme de bases de données (oracle ou non oracle).

5.1 L'outil SQL*Plus

Outil ligne de commande nommé SQLPLUS.

```
|SQLPLUS [ connexion ] [ @fichier_script [argument [,...]] ]
```

Il permet de saisir et d'exécuter des ordres SQL ou du code PL/SQL et dispose en plus d'un certain nombre de commandes.

```
| • sans connexion
| C:\> SQLPLUS /NOLOG
| • avec connexion
| C:\> SQLPLUS system/tahiti@tahiti
| • avec connexion et lancement d'un script sur la ligne de commande
| C:\> SQLPLUS system/tahiti@tahiti @info.sql
|
| • sous dos -----
| set ORACLE_SID=TAHITI
| • connexion sans fichier de mots de passe
| SQL> connect /as sysdba
| Connecté.
|
| SQL> show user
| USER est « SYS »
|
| • sous unix -----
| Export ORACLE_SID=TAHITI
| • Connexion avec un fichier de mots de passe
| SQL> connect sys/secret as sysdba
| Connecté.
|
| SQL> show user
| USER est "SYS"
| SQL>
```



5.1.1 Environnement de travail

SQL*PLUS est avant tout un « interpréteur » de commandes SQL. Il est également fortement interfacé avec le système d'exploitation. Par exemple, sous UNIX, on pourra lancer des commandes UNIX sans quitter sa session SQL*PLUS.

Un SGBDR est une application qui fonctionne sur un système d'exploitation donné. Par conséquent, il faut se connecter au système avant d'ouvrir une session ORACLE. Cette connexion peut être implicite ou explicite.

Pour lancer SQL Plus sans se connecter à une base de données utilisez la commande :

```
C:\> sqlplus /nolog
```

5.1.2 Quelques commandes SQL*Plus

SQL*Plus est un outil composé de commandes de mise en forme et d'affichage :

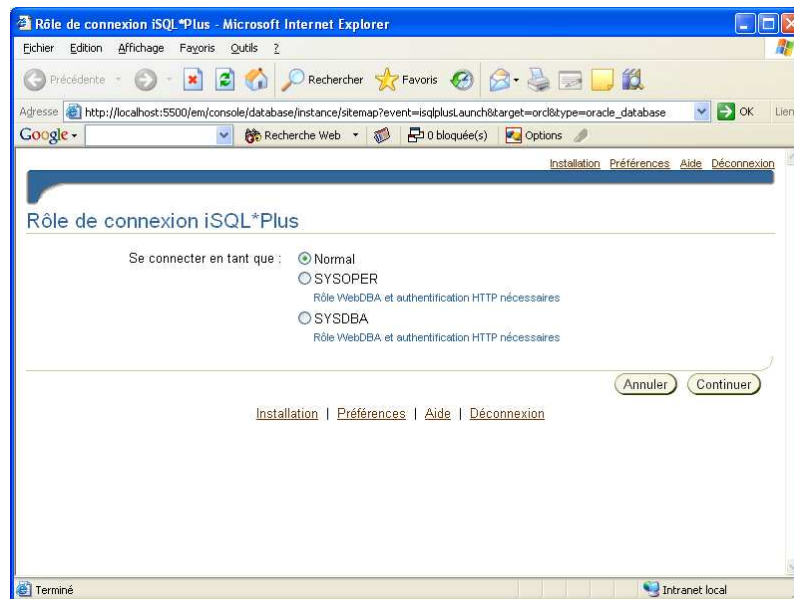
- ◆ COL ADRESSE FORMAT A20, formater l'affichage d'une colonne ADRESSE sur 20 caractères
- ◆ COL PRIXUNIT FORMAT 99.99, formater l'affichage d'une colonne PRIXUNIT
- ◆ CLEAR COL, ré-initialiser la taille des colonnes par défaut
- ◆ SET LINESIZE 100, reformater la taille de la ligne à 100 caractères
- ◆ SET PAUSE ON, afficher un résultat page par page
- ◆ SHOW USER, visualiser le user sous lequel on est connecté
- ◆ CONNECT , se connecter à l'instance
- ◆ [User/MotPass@adresseServeur](#) , permet de changer de session utilisateur
- ◆ CLEAR SCREEN, ré-initialiser l'écran
- ◆ SET SQLPROMPT TEST> , afficher le prompt SQL en : TEST>
- ◆ DESC Nom_Table, afficher la structure d'une table ou d'une vue
- ◆ SPOOL nomfichier.txt, permet d'activer un fichier de format texte dans lequel on retrouvera les commandes et résultats affichés dans SQL Plus
- ◆ SPOOL OFF, permet de désactiver le spool ouvert précédemment
- ◆ @ nom_fichier, permet d'exécuter le contenu d'un fichier sql
- ◆ /, ré-active la dernière commande
- ◆ SET ECHO ON/OFF, affiche ou non le texte de la requête ou de la commande à exécuter
- ◆ SAVE nom_fichier [append|create|replace] , permet de sauvegarder le contenu du buffer courant dans un fichier « .sql »
- ◆ TIMING ON|OFF, provoque l'affichage d'informations sur le temps écoulé, le nombre d'E/S après chaque requête
- ◆ TI ON|OFF, provoque l'affichage de l'heure avec l'invite de commande
- ◆ TERM [ON|OFF] , supprime tout l'affichage sur le terminal lors de l'exécution d'un fichier



- ♦ `VER [ON|OFF]` , provoque l'affichage des lignes de commandes avant et après chaque substitution de paramètre
- ♦ `SQL }` , spécifie le caractère « `}` » comme étant le caractère de continuation d'une commande SQL*Plus
- ♦ `SUFFIX txt` , spécifie l'extension par défaut des fichiers de commande SQL*Plus

5.2 L'outil iSQL*Plus

Outil Internet d'accès à une base de données Oracle, permettant d'écrire des requêtes SQL (d'une façon plus ou moins graphique).



Par défaut, seule la connexion en tant qu'utilisateur « normal » (non SYSDBA ou SYSOPER) est autorisée. Par contre, la connexion en tant qu'utilisateur SYSDBA ou SYSOPER est protégée par une authentification au niveau du serveur HTTP

Pour l'autoriser, il faut au choix :

- ♦ Ajouter des entrées (utilisateur / mot de passe) à l'aide de l'utilitaire `htpasswd` dans un fichier d'authentification du serveur HTTP (défini par défaut dans le fichier de configuration `isqlplus.conf` à : `ORACLE_HOME\sqlplus\admin\iplusdba.pw`)
- ♦ Désactiver l'authentification du serveur HTTP pour ce type de connexion (directive `<Location /isqlplusdba>` dans le fichier de configuration `isqlplus.conf`)

Lors d'une connexion SYSDBA ou SYSOPER, l'URL est modifiée en :

⇒ [http://serveur\[:port\]/isqlplusdba](http://serveur[:port]/isqlplusdba)



5.3 Le Database Control et le Grid control

À partir de la version 10g la base de données Oracle s'est dirigée vers le WEB pour fournir une nouvelle version d'Entreprise Manager à la place de celui de la 9i basé sur java possédant une apparence Windows; ainsi que des variantes selon l'utilisation Dbcontrol pour une seule Base de données ou grid control pour centraliser la gestion de plusieurs bases cibles.

Le *Grid Control* est la console graphique qui permet d'administrer un ensemble de bases de données sur des serveurs distants, appelé « ferme de bases de données ».

Le *Database Contrôle* est en réalité un sous ensemble du Grid Control, correspondant à l'administration de la base de données choisie.

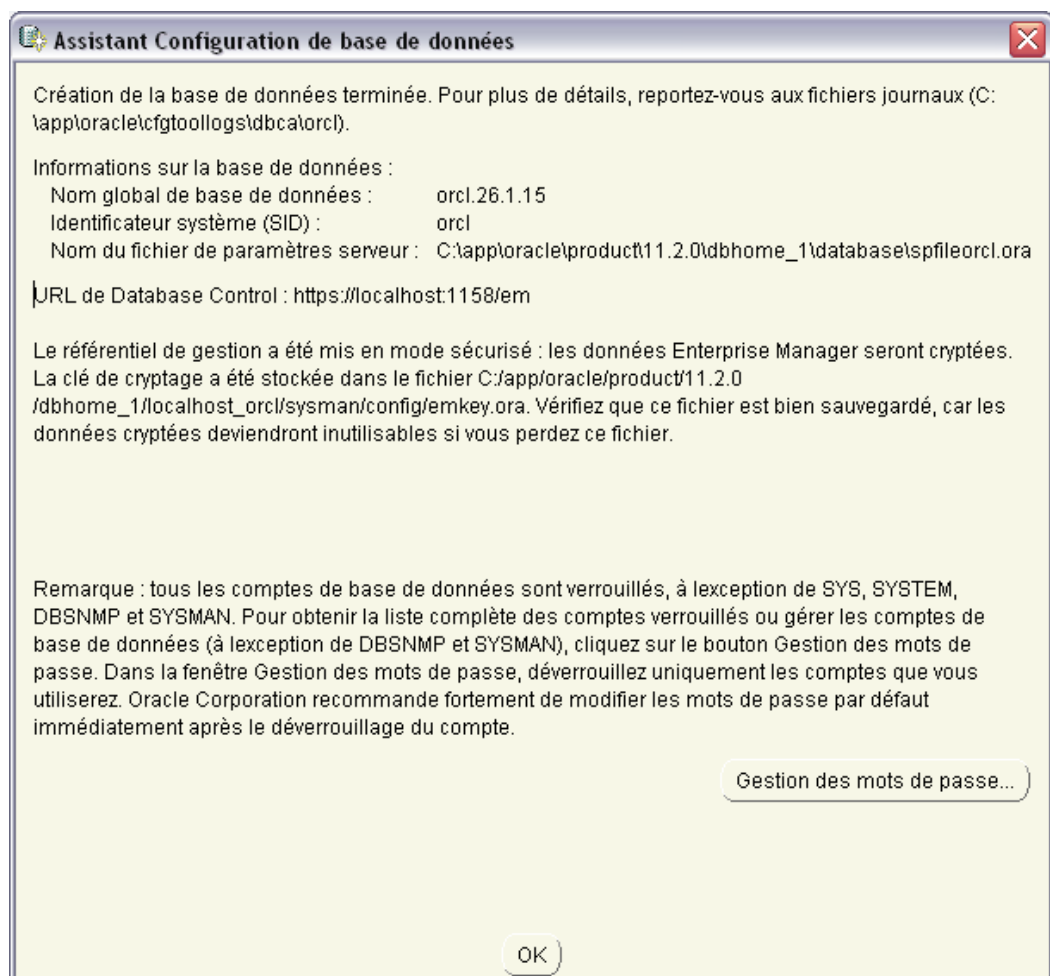
Contrairement au *Grid Control*, le *Database Control* est inclus dans l'installation standard.

Il contient un référentiel et est créé après la création de la base de données.

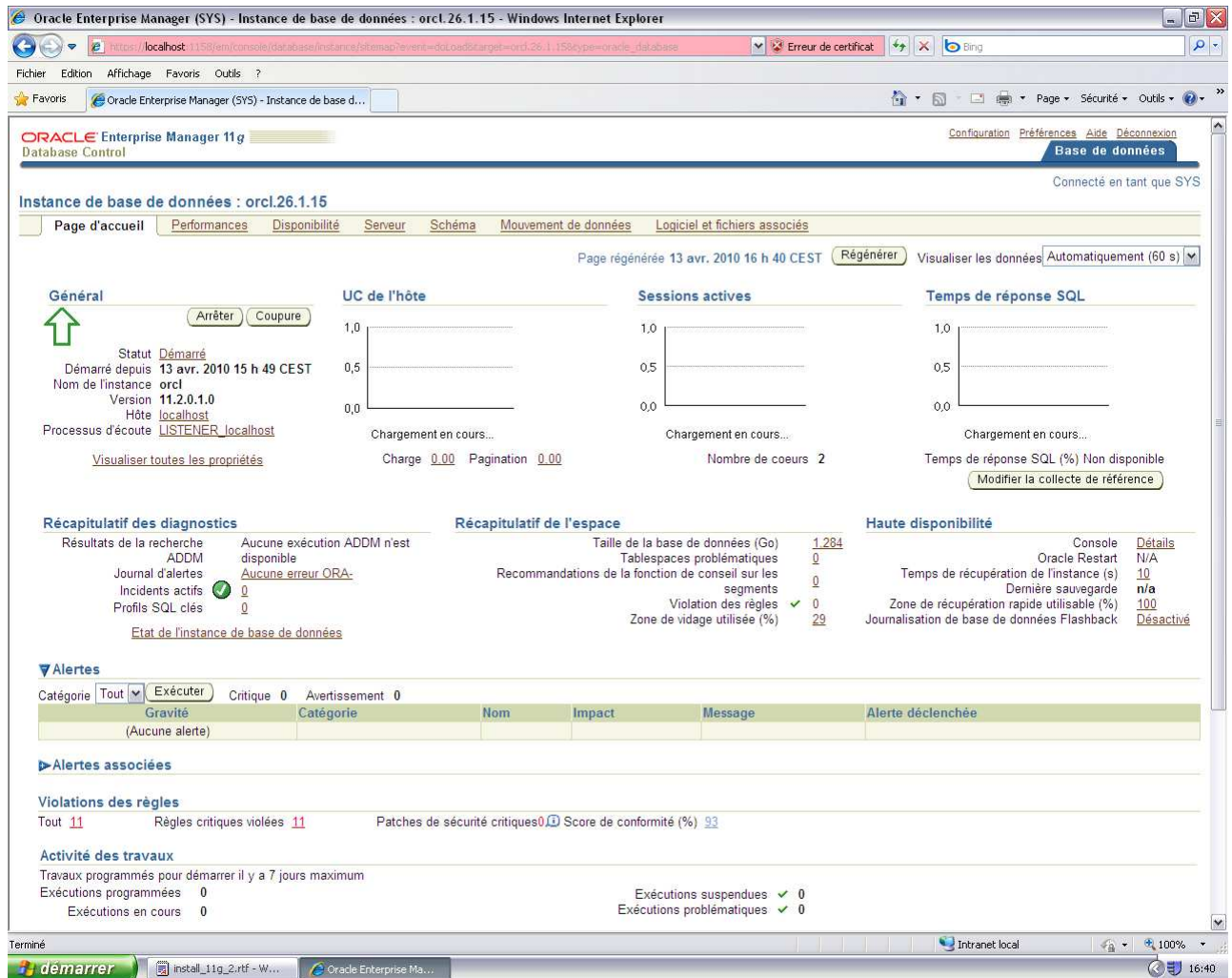
Cette console permet d'administrer directement la base de données :

- ♦ Arrêt/Démarrage, gestion du stockage, gestion des utilisateurs, gestion des schémas, ...
- ♦ Remontée d'alerte, de planification de tâche, de sauvegarde/restauration, d'export/import, ...

Après création d'une base de données Oracle, le *Database Control* peut être affiché sur demande dans le navigateur.



Présentation du database control



En cas de besoin, l'utilitaire *Enterprise Manager Configuration Assistant* (EMCA) peut être utilisé pour créer l'environnement du Database Control pour cette nouvelle base.

```
EMCA [ -r | -x <SID> ]
```

- Sans option l'utilitaire crée l'environnement complet du Database Control.
- -r le référentiel n'est pas créé
- -x <SID> supprime l'environnement du Database control

Si vous utilisez successivement [-x] puis [-r] vous pouvez recréer l'environnement tout en conservant le référentiel existant.

```
D:\cours_Admin10G>emca -x TAHITI
EMCA démarré le Sat Mar 19 12:57:58 CET 2005
La configuration d'Enterprise Manager a réussi.
EMCA arrêté le Sat Mar 19 12:57:58 CET 2005
```



Configuration du database contrôle en fin de création de la base de données

```
connect "SYS"/"&&sysPassword" as SYSDBA
startup ;
host C:\app\oracle\product\11.2.0\dbhome_1\bin\emca.bat -config dbcontrol db -silent -
DB UNIQUE_NAME tahiti -PORT 1521 -EM_HOME C:\app\oracle\product\11.2.0\dbhome_1 -LISTENER
LISTENER -SERVICE_NAME tahiti -SID tahiti -ORACLE_HOME
C:\app\oracle\product\11.2.0\dbhome_1 -HOST localhost -LISTENER_OH
C:\app\oracle\product\11.2.0\dbhome_1 -LOG_FILE
C:\app\oracle\admin\tahiti\scripts\emConfig.log;
spool off
```



6 L'architecture OFA

OFA, *Oracle Flexible Architecture*, est un ensemble de recommandations sur l'arborescence et le nommage des fichiers du serveur contenant les produits et les bases de données en tenant compte de la possibilité d'avoir plusieurs bases de données et plusieurs versions d'Oracle par plate-forme

Un des avantages est de séparer les produits Oracle des fichiers des bases de données.

La norme de la version 11g est présentée page suivante.

Le répertoire `/app/oracle/oradata/orcl/` contient les fichiers de la base de données « orcl »

Le répertoire `/app/oracle/admin/orcl/` contient les répertoires destinés aux exports Data Pump ou non de la base de données ainsi qu'au fichier de paramètre utilisé lors de la création de la base de données « orcl ».

- ⇒ `/app/oracle/admin/orcl/`
 - ⇒ `Adump`
 - ⇒ `Dpdump`
 - ⇒ `pfile`

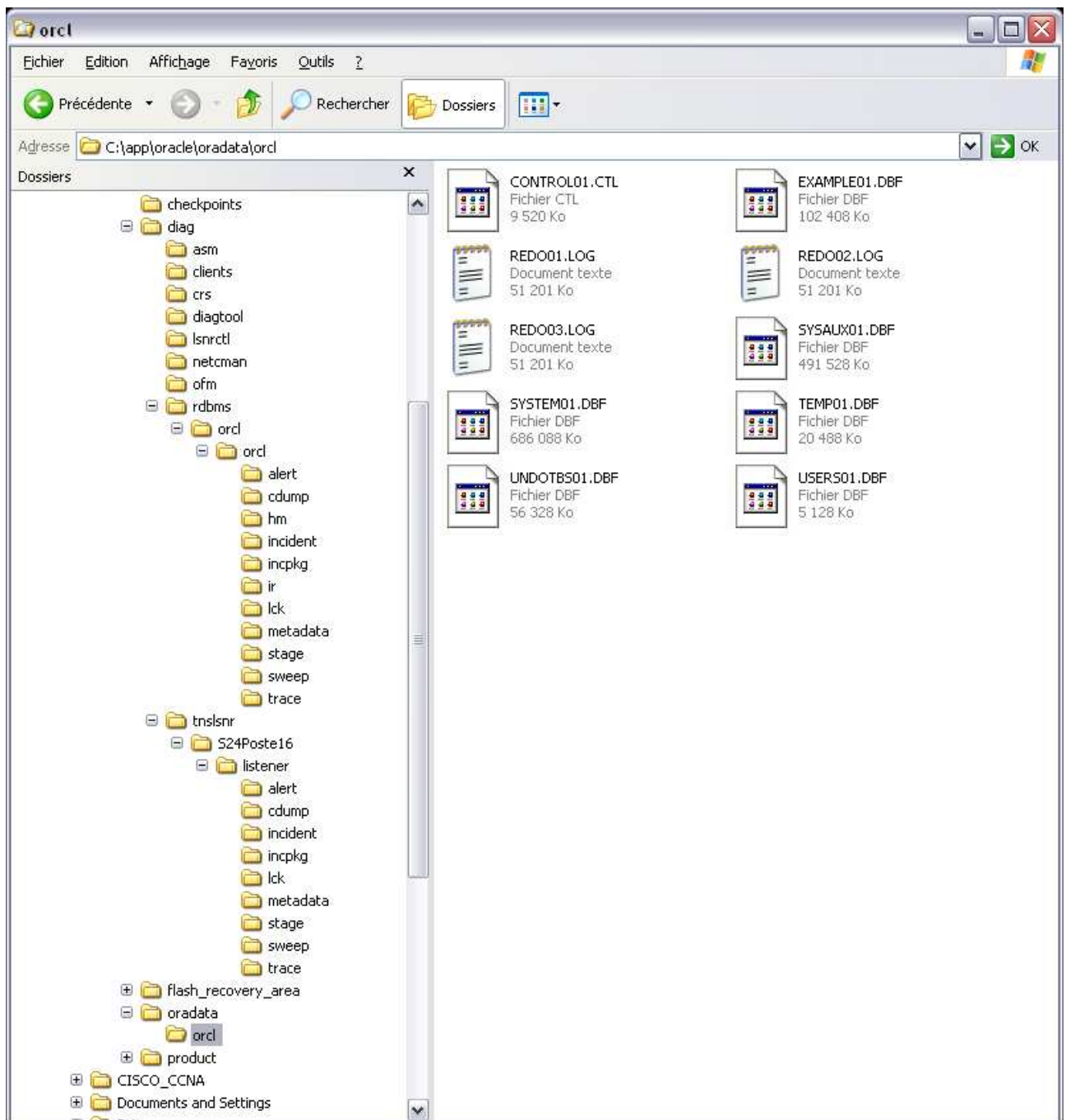
le répertoire `/app/oracle/diag/` contient les répertoires

- ⇒ `/app/oracle/diag/rdbms/orcl/orcl/`
 - ⇒ `Alert` dans lequel est stocké le fichier des alertes en format xml
 - ⇒ `cdump`
 - ⇒ `hm`
 - ⇒ `incident`
 - ⇒ `incpkg`
 - ⇒ `ir`
 - ⇒ `lck` contient un ensemble de fichiers vide représentant des locks
 - ⇒ `metadata` contient un ensemble de fichiers binaires « .ams »
 - ⇒ `stage`
 - ⇒ `sweep`
 - ⇒ `trace` contient un ensemble de fichiers de traces de l'instance
- ⇒ `/app/oracle/flash_recovery_area/orcl/` contient les fichiers de controle multiplexes, et un repertoire ONLINELOG destine aux fichiers de flashback.



Le répertoire `/app/oracle/product/11.2.0/dbhome_1` contient les répertoires des binaires d'oracle. On y retrouve les répertoires

- ⇒ BIN qui contient les binaires d'oracle et certains outils comme « sqlplus.exe ».
- ⇒ Database qui contient sous Windows les fichiers de mot de passe et SPFILE ainsi qu'un sous répertoire d'archive de Redo Log lorsque l'archivage est activé
- ⇒ Dbs qui contient sous unix, les fichiers de mot de passe et SPFILE ainsi qu'un sous répertoire d'archive de Redo Log lorsque l'archivage est activé
- ⇒ NETWORK qui contient le listener et tnsnames.ora



7 Installation Oracle

L'installateur OUI (*Oracle Universal Installer*) est un outil d'installation Oracle compatible OFA (*Oracle flexible Architecture*).

7.1 Pré-requis matériel

Une installation standard peut être effectuée sur une machine avec 1 Go de RAM et 1 Go de swap (mémoire virtuelle) en supplément vous pouvez utiliser le produit avec des composants supérieurs.

Selon votre activité, quand vous installez Oracle, l'installation standard peut être effectuée en moins de 20 minutes.

Sous Unix, bien suivre les pré-requis demandés pour chaque version Unix.

Il faut toujours se référer à la documentation Oracle spécifique à la plate-forme.

⇒ [Installation Guide & Release Notes](#)

Un écran de synthèse est affiché, permettant de vérifier l'installation.

L'installation de Oracle Database 11g automatise la plupart des vérifications de pré-requis pour l'installation.

Si vous choisissez de créer une base de données pendant l'installation d'Oracle, vous devrez répondre à quelques questions permettant de configurer votre base de données.

- ♦ Nom de la base de données, par défaut = ORCL
- ♦ Jeu de caractères à définir
- ♦ ...



A partir de la version 11g, la casse utilisée pour les mots de passe est sensitive !



Programme d'installation Oracle Database 11g version 2 - Installation de la base de données - Etape 4 sur 8

Configuration d'installation standard

Effectue l'installation complète d'Oracle Database avec la configuration de base.

Répertoire de base Oracle Base : C:\app\oracle Parcourir

Emplacement du logiciel : C:\app\oracle\product11.2.0\dbhome_1 Parcourir

Emplacement des fichiers de base de données : C:\app\oracle\oradata Parcourir

Edition de base de données : Enterprise Edition (3,27GB)

Jeu de caractères : Valeur par défaut (WE8MSWIN1252)

Nom global de base de données : orcl.26.1.15

Mot de passe d'administration : [masqué]

Confirmer le mot de passe : [masqué]

Mot de passe de la base de données de départ
Vérifiez que le mot de passe est sécurisé, qu'il contient entre 8 et 128 caractères ...

Messages :

Mot de passe d'administration : [INS-30011] Le mot de passe ADMIN entré n'est pas conforme aux normes recommandées par Oracle.

Aide < Précédent Suivant > Fin Annuler

Programme d'installation Oracle Database 11g version 2 - Installation de la base de données - Etape 6 sur 8

Récapitulatif

Programme d'installation Oracle Database 11g version 2

- Paramètres globaux
 - Espace disque: requis : 3,27 GB ; disponible : 57,13 GB
 - Emplacement source: C:\ora-32\database\install\stage\products.xml
 - Méthode d'installation: Installation standard
 - Edition de base de données: Enterprise Edition (Créer et configurer une base de données)
 - Répertoire de base Oracle Base: C:\app\oracle
 - Emplacement du logiciel: C:\app\oracle\product11.2.0\dbhome_1
- Informations sur l'inventaire
 - Emplacement de l'inventaire: C:\Program Files\Oracle\Inventory
- Informations sur la base de données
 - Configuration: Utilisation générale/Traitement des transactions
 - Nom global de base de données: orcl.26.1.15
 - SID Oracle: orcl
 - Mémoire allouée: 406 Mo
 - Option Gestion automatique de la mémoire: TRUE
 - Jeu de caractères de la base de données : Européen de l'Ouest (WE8MSWIN1252)
 - Méthode de gestion: Database Control
 - Mécanisme de stockage de base de données: Système de fichiers
 - Emplacement des fichiers de base de données: C:\app\oracle\oradata
 - Sauvegarde automatique: Désactivé

Enregistrer le fichier de réponses...

Aide < Précédent Suivant > Fin Annuler



7.2 Installation du client

Cette installation permet d'installer, au minimum, les fichiers nécessaires pour accéder à une base Oracle du réseau (*Couche Oracle Net*).

L'installation d'un client Oracle peut intégrer également :

- ◆ Des outils d'interrogation ou d'administration
- ◆ Des produits pour le développement

L'installation s'effectue avec OUI (*Oracle Universal Installer*) selon les principales étapes suivantes :

- ◆ Désignation de l'emplacement de l'installation (*Oracle Home*)
- ◆ Type d'installation
 - Administrateur, installation de tous les composants
 - Runtime, qui ne contient que Oracle Net, SQL*Plus et les drivers JDBC),
 - Instant Client, ou « client instantané, installation minimale qui ne propose que les « bibliothèques » nécessaires aux applications OCI (*Oracle Call Interface*)
 - Personnalisé permet de choisir les composants à installer.
- ◆ Affichage d'un écran de synthèse permettant de confirmer l'installation

Rappel

L'OCI (Oracle Call Interface) est une application de programmation d'interface (API) qui permet à un développeur d'applications d'utiliser une procédure naturelle, d'un langage de troisième génération ou d'appels de fonctions, pour avoir accès au serveur de base de données d'Oracle pour contrôler toutes les phases de l'exécution de l'expression de SQL. OCI fournit une bibliothèque standard de bases de données et des fonctions de recherche sous la forme de bibliothèques dynamiques en phase d'exécution, ORA, DLL qui peuvent être liées par l'application.



8 Architecture Oracle

L'architecture oracle est constituée d'une instance et d'une base de données appelée database.

Une instance est constituée :

- ⇒ D'une zone de mémoire partagée appelée System Global Area (SGA)
- ⇒ D'un ensemble de processus d'arrière plan ayant chacun un rôle bien précis
- ⇒ D'un ensemble de processus serveur chargés de traiter les requêtes des utilisateurs

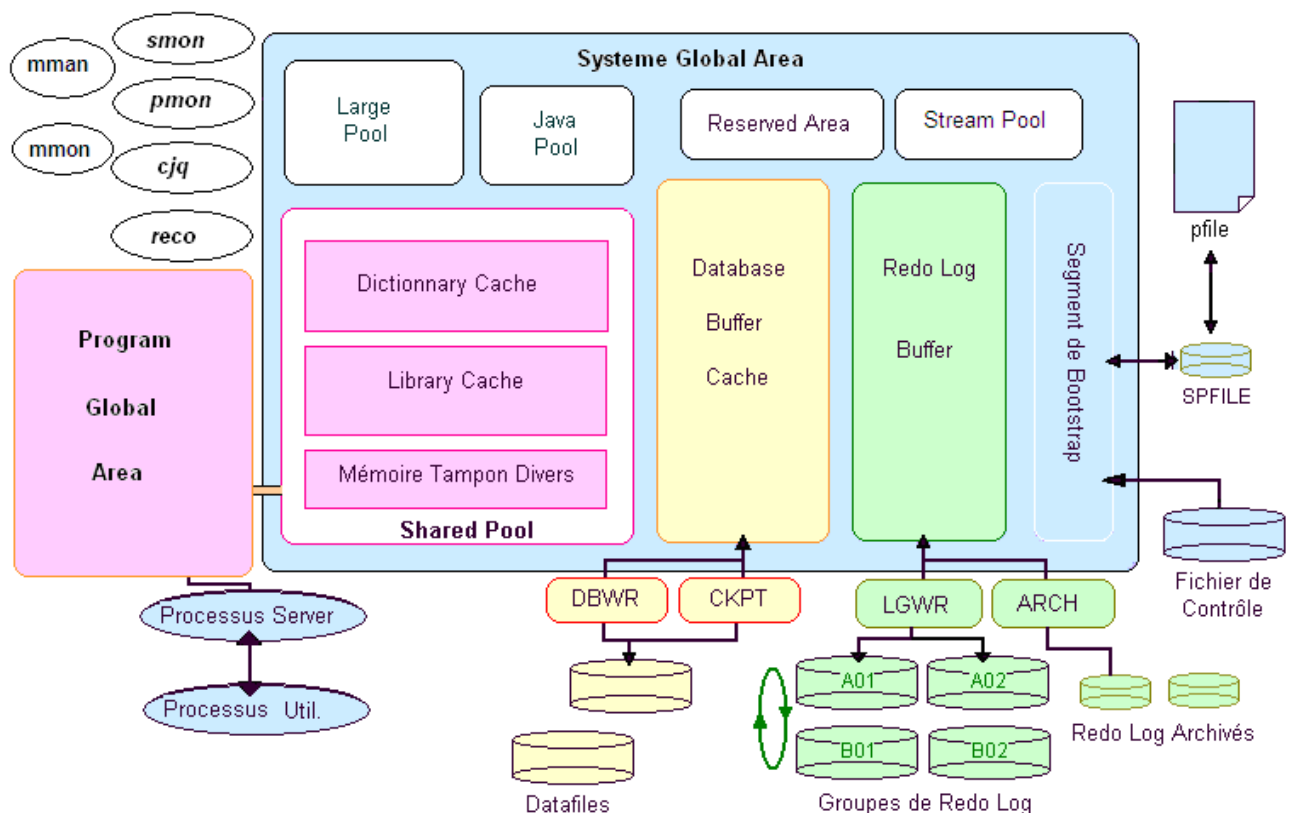
La base de données est l'ensemble des fichiers qui permettent de gérer les données de la base.

Une base de données est constituée de :

- ⇒ Un fichier de contrôle, contenant les informations sur tous les autres fichiers de la base (nom, emplacement, taille).
- ⇒ Fichiers de Redo Log, contenant l'activité des sessions connectées à la base. Ce sont des journaux de transactions de la base. Ils sont organisés en groupe possédant le même nombre de membres.

Et éventuellement, de fichiers de Redo Log archivés contenant les archives d'anciens fichiers de Redo Log.

- ⇒ D'un ou plusieurs fichiers de données qui contiennent les données des tables de la base.



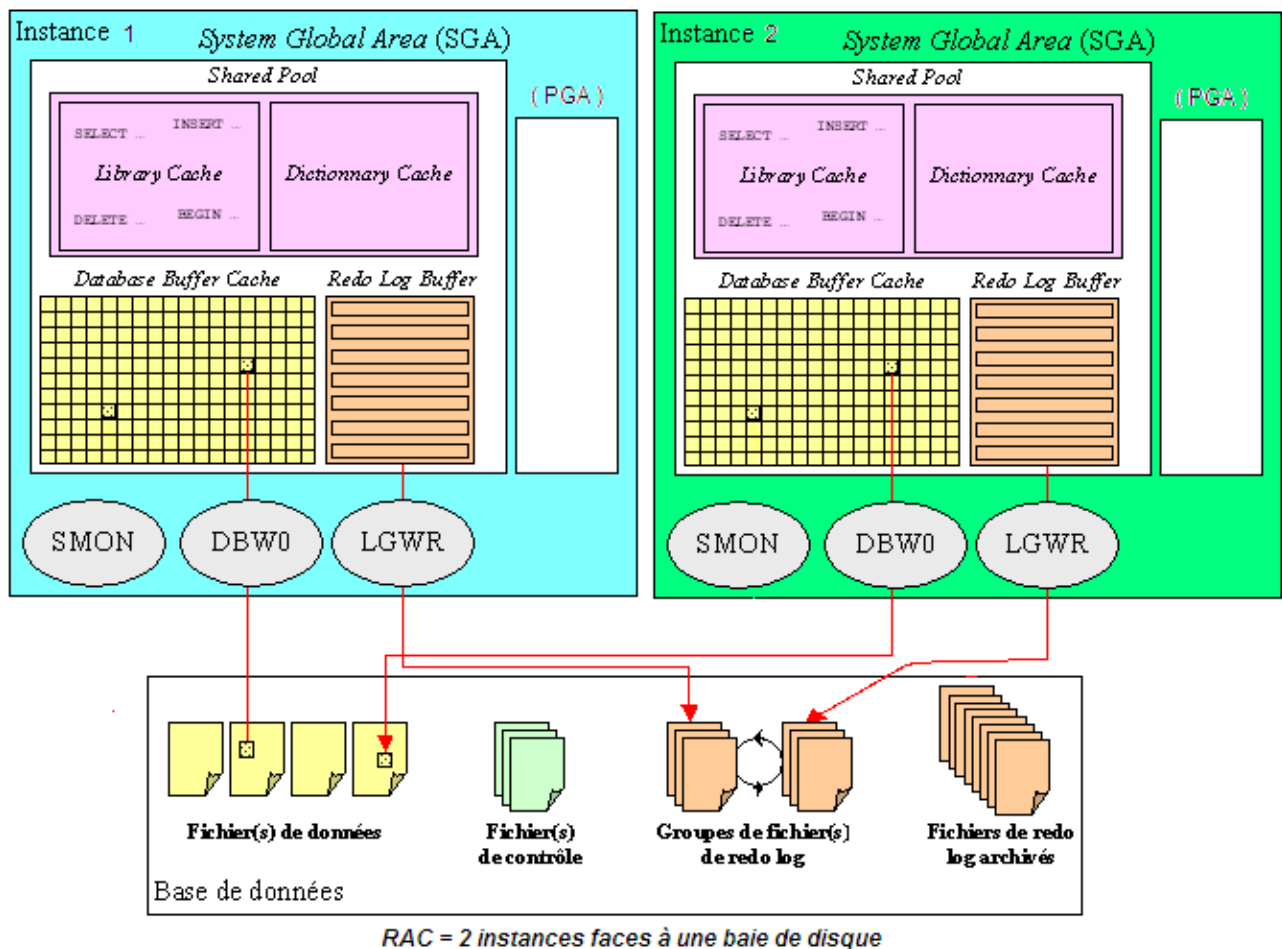
Architecture Interne d'Oracle



Une instance est l'ensemble des processus d'arrière-plan (*background process*) et de zones mémoire qui sont allouées au démarrage de la base de données, pour permettre l'exploitation des données.

Une instance ne peut ouvrir qu'une seule base de données à la fois et dans la grande majorité des cas, une base de données est ouverte par une seule instance.

Néanmoins, moyennant la mise en œuvre de l'option RAC (Oracle Real Application Clusters), permettant d'utiliser Oracle sur des serveurs en cluster, une base de données peut être ouverte par plusieurs instances situées sur des nœuds distincts d'un cluster de serveurs ; cette option est intéressante pour la haute disponibilité mais elle est relativement complexe à mettre en œuvre.



En dehors des processus de l'instance, il existe des processus utilisateurs correspondant à l'application utilisée par l'utilisateur pour se connecter à la base de données (SQL*Plus, un progiciel, un logiciel spécifique, ...).

Dans une architecture client/serveur, ces processus utilisateurs sont situés sur le poste de l'utilisateur et communiquent avec le serveur à travers le réseau grâce à la couche *Oracle Net*.



8.1 Connexion utilisateur

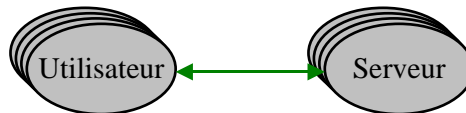
Lorsqu'un utilisateur se connecte à la base de données, il ouvre une session.

Les processus utilisateur sont alors pris en charge par les Processus serveur qui sont chargés de traiter les requêtes des utilisateurs, notamment de charger dans le Database Buffer Cache les données nécessaires.

Le processus serveur communique (localement ou à travers le réseau) avec un processus utilisateur correspondant à l'application de l'utilisateur.

Dans la configuration par défaut, Oracle lance un processus serveur dédié à chaque utilisateur (*dedicated server configuration*)

Mais Oracle peut être configuré en *multithreaded server* (MTS) de manière à avoir des processus serveur partagés par plusieurs processus utilisateur.



L'instance

L'instance est dimensionnée par un ensemble de paramètres stockés dans le fichier de paramètres système SPFILE<SID>.ora, celui-ci a été créé à la création de la base de données à partir d'un fichier de paramètres caractère : PFILE<SID>.ora.

⇒ <SID> correspond au nom de l'instance

8.1.1 La PGA (Program Global Area)

Mémoire privée des différents processus distribuée au moment de la connexion d'un client.

Pour un processus serveur, la PGA contient :

- ⇒ Une zone de tri (allouée dynamiquement lors d'un tri)
- ⇒ Des informations sur la session
- ⇒ Des informations sur le traitement des requêtes de la session
- ⇒ Les variables de session

Dans une configuration multithreaded, une partie de la PGA est en fait stockée dans la SGA (dans la *Large Pool* ou éventuellement dans la *Shared Pool*).

A partir de la version 9i, la PGA devient dynamique et est configurée par le paramètre PGA_AGGREGATE_TARGET.

8.1.2 La SGA: System Global Area

Cette zone de mémoire partagée par les différents processus de l'instance est allouée au démarrage de l'instance et est libérée lors de l'arrêt de celle-ci.



Les principaux composants de la SGA sont :

- ⇒ **SPA : Shared Pool Area** : zone de partage des requêtes et du dictionnaire Oracle.
La *Shared Pool Area* est la partie de la SGA qui est utilisée par Oracle pour partager les requêtes (*Library Cache*) et le dictionnaire de données (*Dictionary Cache*) entre les différents processus.
La Shared Pool est globalement dimensionnée par le paramètre `SHARED_POOL_SIZE` ; la répartition entre le Library Cache et le Dictionary Cache est assurée par Oracle.
Dimensionnée par le paramètre `SHARED_POOL_SIZE`.
- ⇒ **Database Buffer Cache** : Le Database Buffer Cache est un cache de données qui joue le même rôle que la Shared Pool mais pour les données de la base.
Les données de la base ne sont accessibles, en lecture ou en mise à jour, qu'après avoir été chargées dans le Database Buffer Cache.
Dans la pratique, le Database Buffer Cache ayant une taille finie, Oracle utilise un algorithme LRU (*Least Recently Used*) pour gérer le cache : en cas de manque de place, Oracle supprime du cache les données utilisées le moins récemment.
Généralement, augmenter la taille du Database Buffer Cache améliore les performances.
La taille du bloc (`DB_BLOCK_SIZE`) étant fixée à la création de la base, la taille du Database Buffer Cache est définie par la valeur du paramètre `DB_BLOCK_BUFFERS` qui fixe le nombre de buffers en mémoire, chaque buffer ayant une taille égale à `DB_BLOCK_SIZE`.
Le paramètre `DB_BLOCK_BUFFERS` est typiquement compris entre un millier (pour une petite base de test) et plusieurs dizaines/centaines de milliers d'octets.
Dimensionné par le paramètre `DB_CACHE_SIZE`.
- ⇒ **Redo Log Buffer** : Le Redo Log Buffer stocke les informations sur les modifications apportées à la base, avant leur écriture dans un fichier de Redo Log.
L'écriture dans le Redo Log Buffer est séquentielle (les modifications de plusieurs transactions se mélangent) et circulaire (quand le buffer est plein, il repart au début ... après avoir été écrit sur disque dans les fichiers de Redo Log).
Dimensionné par le paramètre `LOG_BUFFER`.
- ⇒ **Large Pool** (à partir de la Version 8), Ajouté en version 8 à l'extérieur du pool partagé pour procurer de l'espace spécifique aux opérations parallèles, à l'usage de la configuration MTS et du module RMAN. En version 10g, la mise en œuvre de l'ASM utilise le Large Pool. Oracle conseille de le dimensionner à 100 Mo dans ce cas.
Dimensionnée par le paramètre `LARGE_POOL_SIZE`.
- ⇒ **Java Pool** (à partir de la Version 8), zone réservée aux programmes Java.
Dimensionné par le paramètre `JAVA_POOL_SIZE`.
L'installation des composants Java impose que cette zone soit configurée, les instructions Java s'y exécutent.
- ⇒ **Streams Pool** (à partir de la Version 10), zone réservée notamment lors de la réplique de données entre bases de données distantes.
Dimensionné par le paramètre `STREAMS_POOL_SIZE`.
- ⇒ **Reserved Area** (à partir de la version 7.3), zone réservée destinée à l'enregistrement d'objets SQL de grande taille (y compris des packages, des procédures et des fonctions).
Dimensionnée par le paramètre `SHARED_POOL_RESERVED_SIZE`.
- ⇒ **Nouveauté 11g : result_cache**
Ce cache est un nouveau composant de la SGA et est utilisé par Oracle pour initialiser le paramètre `MEMORY_TARGET`.
Par défaut ce paramètre est positionné à une valeur égale à 128K.

Ces différentes zones mémoires sont configurées à l'aide du paramètre contenu dans le fichier de paramètres `SPFILE`.

En dehors de la SGA, chaque processus possède une zone de mémoire privée appelée PGA (*Program Global Area*).





La version 11g, offre la possibilité d'automatiser la gestion de l'instance grâce aux paramètres MEMORY_TARGET et MEMORY_MAX_SIZE.

La vue dynamique V\$MEMORY_TARGET_ADVICE

Cette vue dynamique de performances, permet de suivre l'allocation dynamique et visualiser les différentes valeurs de l'allocation dynamique de la mémoire.

Cette vue contient les colonnes :

- ◆ Memory size : taille réelle de la mémoire totale allouée à l'instance
- ◆ Size_factor : coefficient de taille
- ◆ Estd_db_time : taille de l'instance utilisée en mémoire en moyenne par rapport aux facteurs size-factor et time_factor.
- ◆ Time_factor : coefficient de temps
- ◆ Version :

```
|Select * from v$memory_target_advice ;
```

La vue V\$MEMORY_DYNAMIC_COMPONENTS, permet de visualiser les différentes valeurs de chaque pool, entre autre la shared_pool, le database buffer cache, le large pool, etc ...

```
|Select component, current_size, min_size, max_size  
|from v$memory_dynamic_components ;
```

8.2 Le fichier de paramètres (init.ora ou SPFILE.ORA)

Au démarrage, l'instance lit un fichier de paramètres qui contient des paramètres d'initialisation.

Ce fichier est géré par le DBA.

Les paramètres d'initialisation permettent notamment à l'instance :

- ⇒ D'allouer la mémoire souhaitée aux différentes structures de la SGA
- ⇒ De trouver le nom et l'emplacement des fichiers de contrôle de la base

Il existe 2 types de paramètres, les paramètres dynamiques et les paramètres statiques.

Les paramètres dynamiques sont modifiables sans avoir besoin d'arrêter la base de données.

Les vues V\$SYSTEM_PARAMETER et V\$SYSTEM_PARAMETER2 (idem V\$SYSTEM_PARAMETER avec une mise en forme des paramètres) permettent de connaître la valeur des paramètres de l'instance en cours de fonctionnement.



Règles :

- ♦ Les paramètres sont spécifiés sous la forme `nom_paramètre = valeur`
- ♦ Tous les paramètres sont optionnels et ont une valeur par défaut
- ♦ Des commentaires peuvent être inclus et commencent par le caractère `#`
- ♦ La valeur peut être spécifiée entre des guillemets doubles si elle contient des caractères spéciaux (égal, espace, ...)
- ♦ Les valeurs multiples sont spécifiées entre parenthèses, séparées par des virgules

8.3 Les processus d'arrière plan

Il est important de distinguer les processus d'arrière plan des autres processus.

Ils sont indépendants de la connexion des utilisateurs. Ils sont lancés au démarrage de l'instance et arrêtés lors de l'arrêt de l'instance.

Ils réalisent des opérations sur l'instance et sur la base de données, comme l'écriture des fichiers de données, la récupération de la base de données ou la résolution des erreurs.

Certains processus aident à augmenter les performances globales du système.

Principaux processus :

- ♦ **Database Writer (DBWRn)** : écrit sur disque les données modifiées dans le Database Buffer Cache. Les informations de la base de données manipulées par les sessions transitent par ce cache dédié à cet usage.
- ♦ **Log Writer (LGWR)** : écrit sur disque le contenu du Redo Log Buffer dans les fichiers Redo.
- ♦ **Checkpoint (CKPT)** : enregistre les checkpoints dans l'en-tête des fichiers de données. Lorsque qu'un Checkpoint a lieu, toutes les informations qui se trouvent en mémoire sont enregistrées sur disque à l'emplacement prévu. Cet événement correspond à un « jalon » permettant la restauration des données jusqu'à ce point précis dans le temps. CKPT peut à son tour déclencher DBWR et LGWR.
- ♦ **Process Monitor (PMON)** : chargé du nettoyage lors du plantage d'un processus utilisateur. Il libère les ressources de sessions qui se sont mal terminées.
- ♦ **System Monitor (SMON)** : restauration de l'instance après un arrêt anormal. C'est le gardien de la cohésion des données. Une instance cohérente est établie chaque fois que la base est démarrée.
- 10 ♦ **Job Queue Coordinator (CJQ)** : utilisé par le *Scheduler*, il génère les processus pour exécuter les jobs planifiés qui se trouvent dans la file d'attente interne d'Oracle.
Les utilisateurs peuvent créer des jobs et les soumettre à ce coordinateur.
`JOB_QUEUE_PROCESSES > 0` permet de définir le nombre de jobs soumis en simultané.
- 10 ♦ **Memory Manager (MMAN)** : il agit comme un distributeur de mémoire et coordonne la taille allouée aux différents composants.
- 10 ♦ **Memory Monitor (MMON)** : programme et déclenche ADDM (*L'Automatic Database Diagnostic Monitor*) qui effectue des analyses pour déterminer des problèmes potentiels.

Selon la configuration du serveur, d'autres processus d'arrière plan peuvent être présents :

- ♦ **Archiver (ARCn)** : en base « archivée » il archive des fichiers de *Redo Log* chaque fois qu'un fichier Redo est plein.
- ♦ **Recover (RECO)** : gère les bases de données distribuées.
- ♦ **Dispatcher (Dnnnn)** : présent en serveur partagé.



- ♦ *Global Cache service* (LMS) : présent en option RAC (*Real Application Cluster*).
- ♦ *Job Queue* (SNPn) : processus chargé de rafraîchir les *snapshots* ou d'exécuter périodiquement des tâches programmées avec le package DBMS_JOB.

8.4 La base de données

La base de données est l'ensemble des fichiers qui permettent de gérer les données stockées dans de la base de données.

Une base de données est constituée de :

- ⇒ **Un fichier de contrôle**, contenant les informations sur tous les autres fichiers de la base (nom, emplacement, taille).
- ⇒ **Fichiers de Redo Log**, contenant l'activité des sessions connectées à la base. Ce sont des journaux de transactions de la base. Ils sont organisés en groupe possédant le même nombre de membres.

Et éventuellement, de fichiers de Redo Log archivés contenant les archives d'anciens fichiers de Redo Log.

- ⇒ **D'un ou plusieurs fichiers de données** qui contiennent les données proprement dites, elle contient à la création de la base de données au minimum :
 - ♦ Tablespace `SYSTEM`, contenant le dictionnaire de données.
 - ♦ Tablespace `SYSAUX`, c'est le tablespace auxiliaire du tablespace `SYSTEM` contenant des fonctions Oracle ou des données utilisées par des outils tels que le référentiel d'OEM (Oracle Enterprise Manager), placées avant dans un tablespace `OEM_REPOSITORY`, situées aujourd'hui dans le tablespace `SYSAUX`.
 - ♦ Tablespace Temporaire `TEMP`, récupérant les segments temporaires utilisés par les requêtes SQL de la base de données.
 - ♦ Tablespace `UNDO`, récupérant la version précédente des données en cours de modification par les transactions se déroulant sur la base.
 - ♦ Tablespace `USERS`, tablespace de travail par défaut des utilisateurs.
- ⇒ **Un fichier de paramètres binaire SPFILE<SID>.ORA**, contenant les paramètres de démarrage de l'instance et d'autres valeurs qui déterminent l'environnement dans lequel elle s'exécute.
 - Créé à partir d'un fichier de paramètres caractère (INIT<SID>.ora)
- ⇒ **Un fichier de mots de passe**, contenant le mot de passe du privilège SYSDBA.

8.4.1 Les fichiers de données

Ils contiennent les données proprement dites de la base (tables et index notamment).

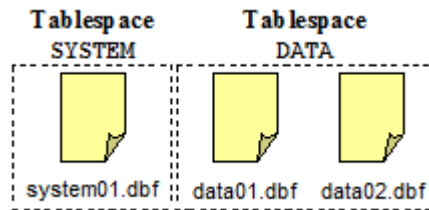
Ils sont logiquement regroupés en tablespaces.

Un tablespace est une unité logique de stockage composée d'un ou plusieurs fichiers physiques.

La quasi totalité des opérations d'administration relatives au stockage s'effectue en travaillant sur le tablespace et non sur le fichier de données.

Dans la pratique, une base comportera donc d'autres fichiers de données appartenant à d'autre tablespaces.



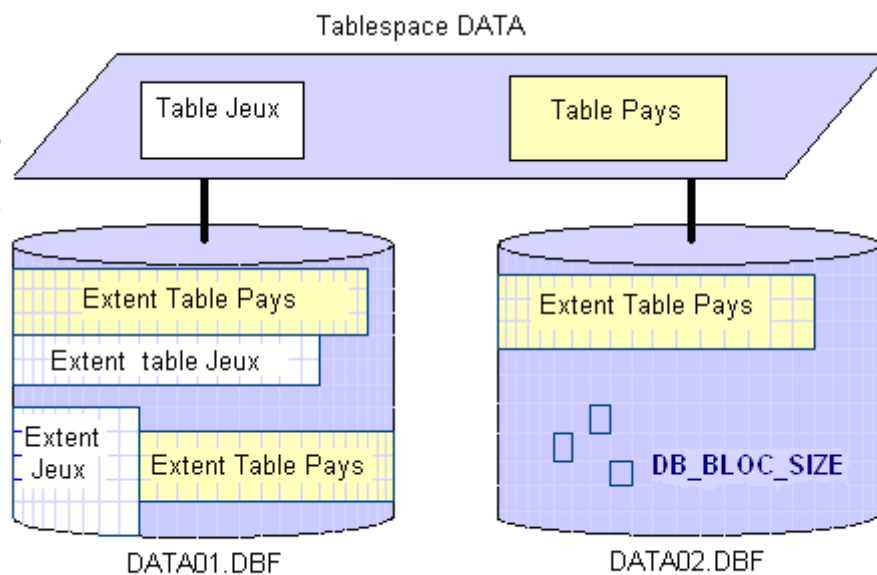


Les fichiers de données sont découpés en blocs d'une taille définie à la création de la base (2 ko, 4 ko, 8 ko, ...). La taille du bloc oracle est définie par le paramètre DB_BLOCK_SIZE.

L'espace occupé par un objet dans un tablespace est désigné par le terme générique de segment.

Un segment appartient à un tablespace et est constitué d'extents.

Un extent est un ensemble de blocs contigus dans un fichier de données.



Notion de Segments et d'Extents

Dans l'image présentée ci-dessus nous voyons que la table Pays objet logique stocké dans le tablespace DATA est constituée de 3 extents ;

- ⇒ 2 extents sont stockés dans le fichier DATA01.DBF
- ⇒ 1 extent est stocké dans le fichier DATA02.DBF.



9 Utilisateurs et connexion à la base de données

A la création d'une base de données un ensemble d'utilisateurs sont créés, dont SYSTEM et SYS.

SYSTEM est l'utilisateur que l'on préférera **pour créer les objets de schéma** tels que les users, les tables ou les index, ... (SYSTEM est un utilisateur qui a des privilèges dba).

L'utilisateur **SYS** (super utilisateur) sera utilisé avec le privilège SYSDBA, pour effectuer des tâches d'administration « lourdes » telles que démarrage ou arrêt de base de données, modification de paramètres systèmes, restauration de base, bref tout ce qui concerne la structure même de la base de données ou de l'instance.



- Utiliser un autre compte SYSTEM pour l'administration courante (objets de schémas).
- Réserver le compte SYS pour les connexions AS SYSDBA
- Ne jamais créer d'objets dans le schéma SYS (autres que ceux du dictionnaire)

9.1 Syntaxe pour la connexion classique

La connexion d'un utilisateur quelconque à une base de données oracle se fait en suivant la syntaxe :

```
|CONNECT utilisateur/mot_de_passe@service_OracleNet
```

```
|SQLPLUS /nolog  
SQL> Connect CHARLY/monpass@bora  
ConnectŮ.  
SQL> Connect SYSTEM/manager@bora  
ConnectŮ.
```

9.2 Syntaxe pour la connexion spéciale SYSDBA ou SYSOPER

Avec une identification par le système d'exploitation

```
|CONNECT / AS { SYSDBA | SYSOPER }
```

```
|$ Export ORACLE_SID=TAHITI  
$ sqlplus /nolog  
  
SQL> Connect /as sysdba  
ConnectŮ.
```



Avec une identification par un fichier de mot de passe

```
|CONNECT utilisateur/mot_de_passe AS { SYSDBA | SYSOPER }
```

```
|SQL> Connect SYS/secret as sysdba  
ConnectŮ.
```

9.3 Les connexions SYSDBA et SYSOPER

SYSDBA : permet toutes les opérations « lourdes » d'administration (création, arrêt, démarrage, restauration, ...).

SYSOPER : même droits que SYSDBA, à l'exception de la création de la base et des restaurations partielles.

Sur un serveur Unix ou Windows, on va vérifier que la variable d'environnement est bien positionnée avant de se connecter à la base de données.



S'assurer que l'instance souhaitée est bien désignée par la variable d'environnement ORACLE_SID, et se connecter en SYSDBA

```
|Sous DOS  
C:\>set oracle_sid=TAHITI  
C:\>sqlplus /nolog  
SQL > CONNECT /AS SYSDBA
```

```
|Sous UNIX  
Export ORACLE_SID=TAHITI  
Echo ORACLE_SID  
TAHITI  
  
SQLPLUS /nolog  
SQL> Connect /as sysdba  
ConnectŮ.
```

9.4 Le fichier de mots de passe

Autrefois créé avec l'utilitaire ORAPWD, il est aujourd'hui créé automatiquement lors de la création de la base de données avec l'outil dbca.

Ce fichier protège le compte SYS associé au privilège SYSDBA permettant une administration lourde (création, démarrage, arrêt, restaurations).

```
|orapwd file=<fichier> password=<mot de passe> [entries=<valeur>]
```




```
rem *** Création du fichier de mots de passe ***
C> REM orapwd
FILE=/app/oracle/product/11.2.0/dbhome_1/database/PWDtahiti.ORA
PASSWORD=secret ENTRIES=10
```

Mettre le paramètre REMOTE_LOGIN_PASSWORDFILE à EXCLUSIVE

Se connecter au système d'exploitation.

Lancer l'outil d'administration et se connecter, en tant que SYS à l'aide du mot de passe défini avec le privilège SYSDBA ou SYSOPER.

```
CONNECT sys/mot_de_passe AS { SYSDBA | SYSOPER }
```

```
SQLPLUS /nolog
SQL> Connect SYS/secret@bora as SYSDBA
ConnectÚ.
```

9.5 Les variables d'environnement

Ces variables doivent être positionnées avant le lancement de l'outil SQL*Plus en mode commande, sous le système d'exploitation.

```
Set ORACLE_SID=orcl
Sqlplus /NOLOG
Connect as sysdba
```

Exemple sous unix

```
export ORACLE_SID=orcl
echo ORACLE_SID
orcl
Sqlplus /NOLOG
Connect as sysdba
```

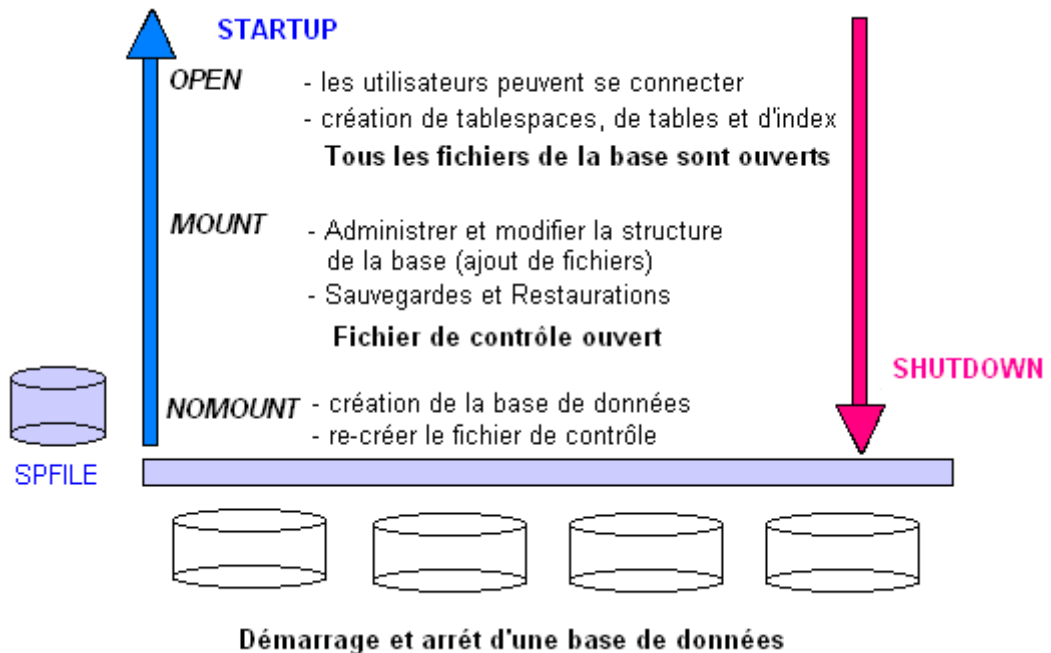
Les principales variables d'environnement sont

- ⇒ **ORACLE_HOME** = définit l'emplacement du noyau Oracle C:\oracle\product\11.2.0\dbhome_1
- ⇒ **ORACLE_BASE** = définit l'emplacement des bases oracle C:\app\oracle
- ⇒ **ORACLE_SID** = désigne le nom de l'instance sur laquelle on veut se positionner
- ⇒ **NLS_LANG** = langage du système d'exploitation FRENCH_FRANCE.WE8MSWIN1252



10 Démarrer & Arrêter une base de données

Une instance peut être démarrée avec 3 niveaux successifs de disponibilité de la base :



Pour rendre une base accessible à tous les utilisateurs, il faut démarrer une instance et ouvrir la base avec cette instance.

Il y a trois étapes dans le processus de démarrage :

- ♦ Démarrage de l'instance
- ♦ Montage de la base
- ♦ Ouverture de la base

Un fichier de paramètres **SPFILE** est lu lors du démarrage de l'instance. Il permet de configurer les paramètres de l'instance.

```
SQL> startup
Instance ORACLE lancée.

Total System Global Area 135338868 bytes
Fixed Size                 453492 bytes
Variable Size             117440512 bytes
Database Buffers          16777216 bytes
Redo Buffers               667648 bytes
Base de données montée.
Base de données ouverte.
SQL>
```



De même, il y a trois étapes dans le processus d'arrêt :

- ◆ Fermeture de la base
- ◆ Démontage de la base
- ◆ Arrêt de l'instance

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

10.1 Démarrer la base de données

Dans SQL*Plus, la commande `STARTUP` permet de démarrer une instance et de lui associer une base de données avec le niveau de disponibilité souhaité.

```
STARTUP [NOMOUNT | MOUNT [nom base] | OPEN [nom_base]]
[RESTRICT] [PFILE=nom_fichier]
;
```

- `NOMOUNT | MOUNT | OPEN` : niveau de disponibilité souhaité
- `nom_base` : nom de la base à monter ou à ouvrir
- `RESTRICT` : restreint l'accès à la base aux utilisateurs ayant le privilège `RESTRICTED SESSION`
- `PFILE` : nom du fichier de paramètres à utiliser



S'assurer que l'instance souhaitée est bien désignée par la variable d'environnement `ORACLE_SID`, et se connecter en `SYSDBA`.

Taper la commande `STARTUP` avec les options souhaitées, puis démarrer une instance sans associer de base (en vue d'en créer une nouvelle ou de recréer le fichier de contrôle) :

- ◆ Démarrer une instance à l'état `MOUNT` pour effectuer certaines tâches d'administration :

```
SQL> startup mount
ORACLE instance started.

Total System Global Area  159383552 bytes
Fixed Size                  788204 bytes
Variable Size              141293844 bytes
Database Buffers           16777216 bytes
Redo Buffers                524288 bytes
Database mounted.
```



♦ Démarrer avec un fichier de paramètres caractère (PFILE)

```
SQL> startup pfile='D:\cours_admin10G\inittahiti02.ora';
ORACLE instance started.

Total System Global Area  159383552 bytes
Fixed Size                  788204 bytes
Variable Size              141293844 bytes
Database Buffers           16777216 bytes
Redo Buffers                524288 bytes
Database mounted.
Database opened.
```

10.2 Modifier la disponibilité de la base de données

Si l'instance a été démarrée dans un niveau intermédiaire (NOMOUNT ou MOUNT), il est possible de la faire passer au niveau supérieur grâce à la commande SQL `ALTER DATABASE` :

♦ NOMOUNT ⇌ MOUNT

```
ALTER DATABASE MOUNT;
```

⇌ MOUNT ⇌ OPEN

```
ALTER DATABASE OPEN;
```



La commande SQL `ALTER DATABASE` ne permet pas de revenir à un niveau inférieur.
Pour cela, il faut arrêter la base et la redémarrer avec le niveau souhaité.

```
SQL> shutdown immediate
Database closed.
Database dismounted.
ORACLE instance shut down.

SQL> startup nomount
ORACLE instance started.

Total System Global Area  159383552 bytes
Fixed Size                  788204 bytes
Variable Size              141293844 bytes
Database Buffers           16777216 bytes
Redo Buffers                524288 bytes
SQL>
SQL> alter database mount;
Base de données modifiée.
SQL> alter database open;
Base de données modifiée.
```



Pour forcer la base à redémarrer vous pouvez utiliser la commande :

```
STARTUP FORCE
```

```
SQL> startup force
ORACLE instance started.

Total System Global Area  159383552 bytes
Fixed Size                  788204 bytes
Variable Size              141293844 bytes
Database Buffers           16777216 bytes
Redo Buffers                524288 bytes
Database mounted.
Database opened.
SQL>
```

10.3 Arrêter la base de données

Dans SQL*Plus, la commande SHUTDOWN permet d'arrêter l'instance et la base de données.

```
SHUTDOWN [NORMAL | IMMEDIATE | TRANSACTIONAL | ABORT]
```

- NORMAL : Oracle attend que tous les utilisateurs soient déconnectés (pas de nouvelle connexion autorisée) puis ferme proprement la base.
- IMMEDIATE : Oracle déconnecte tous les utilisateurs (en effectuant un ROLLBACK des éventuelles transactions en cours) puis ferme proprement la base.
- TRANSACTIONAL : Oracle attend que toutes les transactions en cours se terminent avant de déconnecter les utilisateurs (pas de nouvelle transaction autorisée) puis ferme et démonte proprement la base.
- ABORT : Oracle déconnecte tous les utilisateurs (sans effectuer de ROLLBACK des éventuelles transactions en cours) puis ferme brutalement la base ; une restauration de l'instance sera nécessaire lors du prochain démarrage.

Lancer l'outil d'administration et se connecter AS SYSDBA, en s'assurant que l'instance souhaitée est correctement désignée.

```
SQL> connect /@tahiti as sysdba
ConnectŮ.

SQL> select instance_name from v$instance;
INSTANCE_NAME
Tahiti
```



10.4 Ouvrir la base de données en mode RESTRICT

Pour ouvrir la base en mode restreint, il suffit d'ouvrir la base en précisant la clause : `ENABLE RESTRICTED SESSION`.

Lorsque vous avez placé l'instance en mode `RESTRICTED SESSION` vous pouvez effectuer des tâches d'administration en étant seul connecté.



Pour ouvrir la base en mode `RESTRICT` il faut avoir les privilèges system : `CREATE SESSION` et `RESTRICTED SESSION`

Pour ouvrir l'instance en mode `RESTRICT`, exécutez la commande :

```
STARTUP RESTRICT
```

```
SQL> startup restrict
ORACLE instance started.

Total System Global Area  159383552 bytes
Fixed Size                  788204 bytes
Variable Size              141293844 bytes
Database Buffers           16777216 bytes
Redo Buffers                524288 bytes
Database mounted.
Database opened.

SQL> select instance_name,logins from v$instance;
INSTANCE_NAME      LOGINS
-----
tahiti              RESTRICTED
```

Puis pour repasser l'instance en mode `NORMAL`, utilisez la commande :

```
ALTER SYSTEM DISABLE RESTRICTED SESSION ;
```

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION ;
System altered.
SQL> select instance_name,logins from v$instance;
INSTANCE_NAME      LOGINS
-----
tahiti              ALLOWED
```



10.5 Mettre l'instance dans un état QUIESCE

Oracle 9i permet de mettre l'instance dans un état QUIESCE où seule l'activité de SYS et SYSTEM est autorisée pour réaliser des manipulations sur la base de données en évitant les accès concurrents.

Les autres utilisateurs ne peuvent pas travailler même s'ils possèdent un rôle DBA ou le privilège SYSDBA.

Oracle laisse les sessions actives (requêtes en cours) se terminer avant de passer l'instance dans l'état QUIESCE (ce qui peut être long).

Pendant ce temps, aucune session inactive ne peut devenir active (pas de nouvelle requête autorisée).

Pendant que l'instance est en état QUIESCE, les demandes de connexion ou les nouvelles requêtes sont mises en attente sans message (la session paraît bloquée).

La colonne ACTIVE_STATE de la vue V\$INSTANCE donne l'état de la base de données :

- ⇒ NORMAL = l'instance autorise tous les utilisateurs à travailler.
- ⇒ QUIESCING = l'instance est en train de passer dans l'état QUIESCE, elle attend que les sessions actives deviennent inactives.
- ⇒ QUIESCED = l'instance est dans l'état QUIESCE



Nécessite que la fonctionnalité de gestion des plans de ressource soit activée (*Database Ressource Manager*). Positionner le paramètre RESOURCE_MANAGER_PLAN = nom du plan (INTERNAL_PLAN qui est le plan par défaut).

```
SQL> alter system quiesce restricted;
alter system quiesce restricted
*
ERREUR Ó la ligne 1 :
ORA-25507: le gestionnaire de ressources n'a pas été continuellement actif
```

- Mettre l'instance dans l'état QUIESCE

```
ALTER SYSTEM QUIESCE RESTRICTED;
```

- Arrêt de l'état QUIESCE

```
ALTER SYSTEM UNQUIESCE;
```

10.6 Vues du dictionnaire de données

Au niveau du dictionnaire de données, pour trouver des informations sur les bases identifiées sur un serveur, consultez les vues suivantes qui sont accessibles à un utilisateur de type administrateur.

- V\$INSTANCE : informations sur l'instance
- V\$DATABASE : informations sur la base
- V\$SGA : informations sur la SGA
- V\$PARAMETER : informations sur les paramètres actifs



- V\$VERSION : informations sur la version d'Oracle
- V\$OPTION : informations sur les options disponibles
- DATABASE_PROPERTIES : informations sur les propriétés par défaut de la base de données
- DATABASE_SUMMARY : informations de la base sur les service déclarés, le nom du serveur, et le character set.
- NLS_DATABASE_PARAMETERS : paramètre NLS de la base
- V\$MEMORY_DYNAMIC_COMPONENTS, permet de visualiser les différentes valeurs de chaque pool, entre autre la shared_pool, le database buffer cache, le large pool, etc ...

```
Select component, current_size, min_size, max_size  
from v$memory_dynamic_components ;
```

La vue dynamique V\$MEMORY_TARGET_ADVICE

Cette vue dynamique de performances, permet de suivre l'allocation dynamique et visualiser les différentes valeurs de l'allocation dynamique de la mémoire.

Cette vue contient les colonnes :

- ♦ Memory size : taille réelle de la mémoire totale allouée à l'instance
- ♦ Size_factor : coefficient de taille
- ♦ Estd_db_time : taille de l'instance utilisée en mémoire en moyenne par rapport aux facteurs size-factor et time_factor.
- ♦ Time_factor : coefficient de temps
- ♦ Version :

```
Select * from v$memory_target_advice ;
```



11 Gestion de l'instance et SPFILE

Au démarrage, l'instance lit un fichier de paramètres binaire `SPFILE` qui contient des paramètres d'initialisation. Ce fichier est géré par le DBA.

Il s'agit d'un référentiel centralisé des paramètres d'initialisation de l'instance au démarrage de la base de données en binaire qui permet d'effectuer des modifications de paramètres pendant le fonctionnement de l'instance (sans avoir besoin d'arrêter la base de données).

Règles concernant l'écriture des paramètres :

- ⇒ Les paramètres sont spécifiés sous la forme `nom_paramètre = valeur`
- ⇒ Tous les paramètres sont optionnels et ont une valeur par défaut
- ⇒ Des commentaires peuvent être inclus et commencent par le caractère `#`
- ⇒ La valeur peut être spécifiée entre des guillemets doubles si elle contient des caractères spéciaux (égal, espace, ...)
- ⇒ Les valeurs multiples sont spécifiées entre parenthèses, séparées par des virgules

Ces paramètres sont pris en compte directement en mémoire (paramètres dynamiques) ou uniquement dans le `SPFILE` (paramètres statiques). Dans dernier ce cas il faut arrêter puis redémarrer la base de données pour que la modification soit prise en compte par l'instance.

11.1 Créer le fichier du paramètre SPFILE

Un fichier de paramètres serveur peut être exporté au format texte par l'ordre SQL :

```
CREATE SPFILE [ = 'nom spfile' ] FROM PFILE [ = 'nom pfile' ]  
;
```



Cette action nécessite une connexion SYSDBA ou SYSOPER.

```
• se connecter as sysdba  
SQL> connect /@tahiti as sysdba  
Connecté.  
  
• créer le fichier de paramètres SPFILE  
SQL> create spfile from pfile='d:\tahiti\pfile\inittahiti.ora';  
Fichier créé.
```

Dans l'optique de l'utilisation d'un fichier de paramètres commun à plusieurs instances (par exemple en RAC), ceux-ci peuvent être spécifiés sous la forme : « instance.paramètre », le symbole « * » désignant n'importe quelle instance (`*_SHARDE_POOL_SIZE`).

- ⇒ C'est cette syntaxe qui est utilisée lors de l'export d'un fichier `SPFILE`.



11.2 Exporter un fichier de paramètres serveur SPFILE

Le fichier généré peut être utilisé à des fins de simple consultation ou de modification, pour créer le SPFILE à partir du PFILE (init<SID>.ora) modifié ou pour effectuer des démarrages particuliers.

```
CREATE PFILE [ = 'nom pfile' ] FROM SPFILE [ = 'nom spfile' ]  
;
```

```
• Exporter le fichier de paramètres SPFILE  
SQL> create pfile from spfile ;  
File created.
```

Le fichier « INITOrcl.ORA » est généré dans le répertoire ORACLE_HOME/database sous Windows et dans le répertoire ORACLE_HOME/dbs sous unix.

INITOrcl.ORA

```
orcl._db_cache_size=96468992  
orcl._java_pool_size=4194304  
orcl._large_pool_size=4194304  
orcl._oracle_base='C:\app\oracle'#ORACLE_BASE set from environment  
orcl._pga_aggregate_target=146800640  
orcl._sga_target=281018368  
orcl._shared_io_pool_size=0  
orcl._shared_pool_size=163577856  
orcl._streams_pool_size=4194304  
*.audit_file_dest='C:\app\oracle\admin\orcl\adump'  
*.audit_trail='db'  
*.compatible='11.2.0.0.0'  
*.control_files='C:\app\oracle\oradata\orcl\control01.ctl',  
                'C:\app\oracle\flash_recovery_area\orcl\control02.ctl'  
*.db_block_size=8192  
*.db_domain='26.1.15'  
*.db_name='orcl'  
*.db_recovery_file_dest='C:\app\oracle\flash_recovery_area'  
*.db_recovery_file_dest_size=4039114752  
*.diagnostic_dest='C:\app\oracle'  
*.dispatchers='(PROTOCOL=TCP) (SERVICE=orclXDB)'  
*.local_listener='LISTENER_ORCL'  
*.memory_target=425721856  
*.open_cursors=300  
*.processes=150  
*.remote_login_passwordfile='EXCLUSIVE'  
*.undo_tablespace='UNDOTBS1'
```

Les colonnes ISSUES_MODIFIABLE et ISSYS_MODIFIABLE de la vue V\$PARAMETER donnent des informations sur le type de paramètre.

- ⇒ La colonne ISSUES_MODIFIABLE vaut TRUE ou FALSE selon que le paramètre est modifiable ou non au niveau de la session.
- ⇒ La colonne ISSYS_MODIFIABLE vaut FALSE si le paramètre n'est pas modifiable au niveau du système, et DEFERRED ou IMMEDIATE selon qu'il est modifiable en différé ou immédiatement.



```
SQL> set pagesize 100
SQL> col name format A16
SQL> col value format A40
SQL> select name, value, isses_modifiable, issys_modifiable
2   from v$parameter
3   where name='control_files'
4     or name='shared_pool_size'
5     or name='sort_area_size'
6   order by name;
```

NAME	VALUE	ISSES	ISSYS_MOD
control_files	D:\Oracle\oradata\TAHITI\control01.ctl, D:\Oracle\oradata\TAHITI\control02.ctl	FALSE	FALSE
shared_pool_size	16777216	FALSE	IMMEDIATE
sort_area_size	65536	TRUE	DEFERRED

11.3 Modifier des paramètres de l'instance ou du SPFILE

L'ordre SQL `ALTER SYSTEM` permet de modifier dynamiquement la valeur des paramètres d'initialisation.

```
ALTER SYSTEM SET paramètre = valeur [...] [ COMMENT = 'texte' ]
[ DEFERRED ] [ SCOPE = MEMORY | SPFILE | BOTH ]
;
```

- Paramètre : nom du paramètre
- Valeur : valeur attribuée au paramètre
- « COMMENT = 'texte' » : commentaire associé à la modification du paramètre. Inséré dans le fichier de paramètres serveur si ce dernier est la cible de la modification (voir la clause SCOPE).
- DEFERRED : si présent, indique que la modification ne concerne que les futures sessions, pas celles actuellement connectées. N'a de sens que si la mémoire est la cible de la modification (voir la clause SCOPE). Peut être obligatoire pour certains paramètres.
- SCOPE : définit la cible de la modification.
- MEMORY : la mémoire seulement
- SPFILE : le fichier de paramètres serveur seulement
- BOTH : les deux

```
• Modification d'un paramètre uniquement en mémoire
SQL> SELECT value FROM v$parameter WHERE name = 'shared_pool_size';
VALUE
167772168

SQL> ALTER SYSTEM SET SHARED_POOL_SIZE = 80M
2   SCOPE = memory;
Système modifié.
```



11.4 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent de visualiser les paramètres :

- V\$PARAMETER = valeur actuelle des paramètres.
- V\$PARAMETER2 = identique à V\$PARAMETER mais avec un affichage sur plusieurs lignes des paramètres qui ont une liste de valeurs (comme le paramètre CONTROL_FILES par exemple).
- V\$SPFILE = contenu actuel du fichier de paramètres serveur actif. (le contenu de la vue est vide si l'instance n'utilise pas de fichier de paramètres serveur). Donne la valeur du paramètre situé dans le SPFILE.
- SHOW parameter SGA : cette commande affiche tous les paramètres contenant le mot SGA dans SQL*Plus

La vue dynamique V\$SYSTEM_PARAMETER

Cette vue dynamique de performances, permet de la valeur des paramètres de l'instance.

Cette vue contient les colonnes :

- ♦ NAME : Nom du paramètre (en minuscule)
- ♦ VALUE : valeur du paramètre
- ♦ DISPLAY_VALUE : valeur du paramètre avec mise en forme à l'affichage
- ♦ ISDEFAULT : TRUE si le paramètre est égal à sa valeur par défaut, FALSE autrement.
- ♦ ISSYS_MODIFIABLE : TRUE si le paramètre n'est pas modifiable au niveau de la session, FALSE sinon
- ♦ ISSYS_MODIFIABLE : FALSE si le paramètre n'est pas modifiable au niveau du système, et DEFERRED s'il est modifiable en différé et IMMEDIATE s'il est modifiable immédiatement.
- ♦ ISMODIFIED : indique si le paramètre a été modifié depuis le démarrage de l'instance.
- ♦ ISDEPRECATED : TRUE si le paramètre est déprécié.



12 Créer une base de données

La naissance d'une base de données Oracle se fait lors de la conception de celle-ci.

Toute erreur à ce niveau verra la base de données affligée par des dégradations de performances importantes. Et souvent seule une nouvelle conception permettra une optimisation réelle de celle-ci.

Le processus complet de création d'une nouvelle base de données pour une application comporte les étapes suivantes :

- ⇒ Conception du modèle conceptuel de données (MCD)
- ⇒ Conception du modèle logique puis physique de données (MLD et MPD)
- ⇒ Création de la base proprement dite (présenté dans ce chapitre)

Les différentes étapes de la création de la base de données proprement dite sont :

- ◆ Créer les répertoires sur les disques
- ◆ Préparer un nouveau fichier de paramètres `init<SID>.ora`
- ◆ Créer un fichier de paramètres serveur à partir du fichier `init<SID>.ora`
- ◆ positionner `ORACLE_SID` , au nom de l'instance
- ◆ Sous Windows uniquement, créer le service associé à l'instance en utilisant l'outil ORADIM (qui gère les services rattachés aux instances des bases oracle)
- ◆ Démarrer l'instance en état `NOMOUNT`
- ◆ Créer la base en exécutant l'ordre `CREATE DATABASE`
- ◆ Installer le dictionnaire de données et packages destinés au bon fonctionnement de la base de données
- ◆ Remplir la base de données avec les objets de schéma destiné aux applications
 - Création des structures de stockage adaptées (tablespaces)
 - Création du compte Oracle qui va contenir les objets de l'application (utilisateur propriétaire des objets applicatifs)
 - Création des objets de l'application dans ce compte Oracle
 - Création des utilisateurs finaux de l'application
 - Sauvegarde de la base de données



A partir de la version 10g, il faut utiliser DBCA pour créer une base de données.

- c'est bien plus facile et surtout sécurisé !
- vous pouvez aussi générer les scripts de création de la base puis les exécuter !



12.1 Présentation du script de création de la base

```
set verify off
ACCEPT sysPassword CHAR PROMPT 'Enter new password for SYS: ' HIDE
ACCEPT systemPassword CHAR PROMPT 'Enter new password for SYSTEM: ' HIDE
ACCEPT sysmanPassword CHAR PROMPT 'Enter new password for SYSMAN: ' HIDE
ACCEPT dbsnmpPassword CHAR PROMPT 'Enter new password for DBSNMP: ' HIDE
host C:\app\oracle\product\11.2.0\dbhome_1\bin\orapwd.exe
file=C:\app\oracle\product\11.2.0\dbhome_1\database\PWDTahiti.ora force=y

OLD_UMASK='umask'
umask 0027
mkdir C:\app\oracle\admin\tahiti\dpdump
mkdir C:\app\oracle\admin\tahiti\pfile
mkdir C:\app\oracle\cfgtoollogs\dbca\tahiti
mkdir C:\app\oracle\flash_recovery_area
mkdir C:\app\oracle\flash_recovery_area\tahiti
mkdir C:\app\oracle\oradata\tahiti
mkdir C:\app\oracle\product\11.2.0\dbhome_1\database
umask ${OLD_UMASK}
set ORACLE_SID=tahiti
set PATH=%ORACLE_HOME%\bin;%PATH%
C:\app\oracle\product\11.2.0\dbhome_1\bin\oradim.exe -new -sid TAHITI -startmode manual -
spfile
C:\app\oracle\product\11.2.0\dbhome_1\bin\oradim.exe -edit -sid TAHITI -startmode auto -
srvstart system
C:\app\oracle\product\11.2.0\dbhome_1\bin\sqlplus /nolog
@C:\app\oracle\admin\tahiti\scripts\tahiti.sql
```

• Creation de la base-

```
SET VERIFY OFF
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\app\oracle\admin\tahiti\scripts\CreatedB.log append
startup nomount pfile="C:\app\oracle\admin\tahiti\scripts\init.ora";
CREATE DATABASE "tahiti"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100
CHARACTER SET AL32UTF8
NATIONAL CHARACTER SET AL16UTF16
USER SYS IDENTIFIED BY "&&sysPassword"
USER SYSTEM IDENTIFIED BY "&&systemPassword"
DATAFILE 'C:\app\oracle\oradata\tahiti\system01.dbf' SIZE 700M REUSE
AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE 'C:\app\oracle\oradata\tahiti\sysaux01.dbf' SIZE 600M REUSE
AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
SMALLFILE DEFAULT TEMPORARY TABLESPACE TEMP TEMPFILE
'C:\app\oracle\oradata\tahiti\temp01.dbf' SIZE 20M REUSE
AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
SMALLFILE UNDO TABLESPACE "UNDOTBS1" DATAFILE 'C:\app\oracle\oradata\tahiti\undotbs01.dbf'
SIZE 200M REUSE
AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
LOGFILE
GROUP 1 ('C:\app\oracle\oradata\tahiti\redo01.log') SIZE 51200K,
GROUP 2 ('C:\app\oracle\oradata\tahiti\redo02.log') SIZE 51200K,
GROUP 3 ('C:\app\oracle\oradata\tahiti\redo03.log') SIZE 51200K
;
spool off
```

• creation du tablespace USERS -

```
SET VERIFY OFF
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
```



```
spool C:\app\oracle\admin\tahiti\scripts\CreateDBFiles.log append
CREATE SMALLFILE TABLESPACE "USERS" LOGGING DATAFILE
'C:\app\oracle\oradata\tahiti\users01.dbf' SIZE 5M REUSE AUTOEXTEND ON NEXT 1280K MAXSIZE
UNLIMITED EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
ALTER DATABASE DEFAULT TABLESPACE "USERS";
spool off
```

• **creation du dictionnaire de données -**

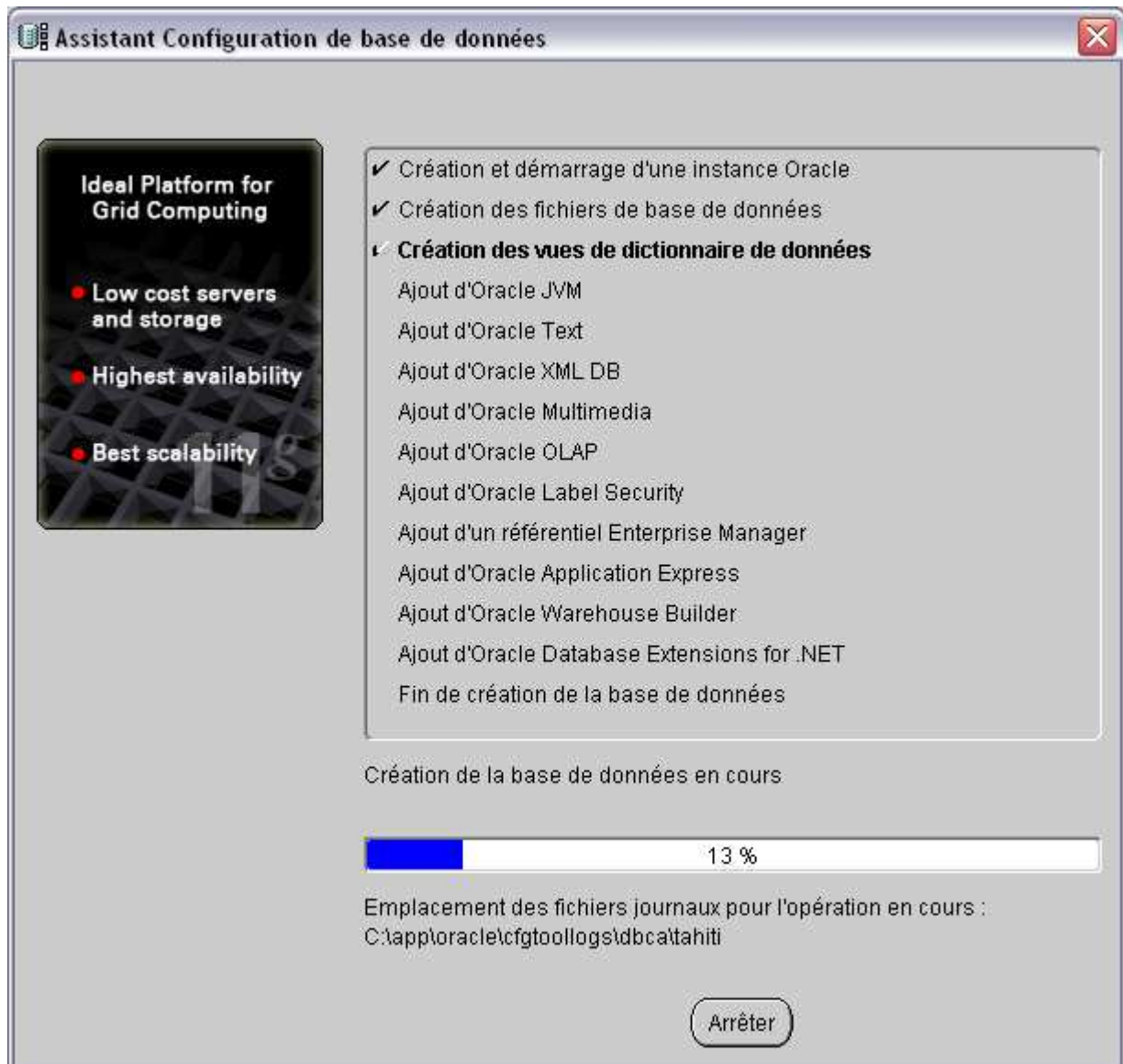
```
SET VERIFY OFF
connect "SYS"/"&&sysPassword" as SYSDBA
set echo on
spool C:\app\oracle\admin\tahiti\scripts\CreateDBCatalog.log append
@C:\app\oracle\product\11.2.0\dbhome_1\rdbms\admin\catalog.sql;
@C:\app\oracle\product\11.2.0\dbhome_1\rdbms\admin\catblock.sql;
@C:\app\oracle\product\11.2.0\dbhome_1\rdbms\admin\catproc.sql;
@C:\app\oracle\product\11.2.0\dbhome_1\rdbms\admin\catoctk.sql;
@C:\app\oracle\product\11.2.0\dbhome_1\rdbms\admin\owminst.plb;
connect "SYSTEM"/"&&systemPassword"
@C:\app\oracle\product\11.2.0\dbhome_1\sqlplus\admin\pupbld.sql;
connect "SYSTEM"/"&&systemPassword"
set echo on
spool C:\app\oracle\admin\tahiti\scripts\sqlPlusHelp.log append
@C:\app\oracle\product\11.2.0\dbhome_1\sqlplus\admin\help\hlpbld.sql helpus.sql;
spool off
```

Après la création de la base de données et du dictionnaire de données vous pouvez installer des modules supplémentaires qui vous permettront de gérer des bases stockant des données relatives à internet ou autorisant le datamining.

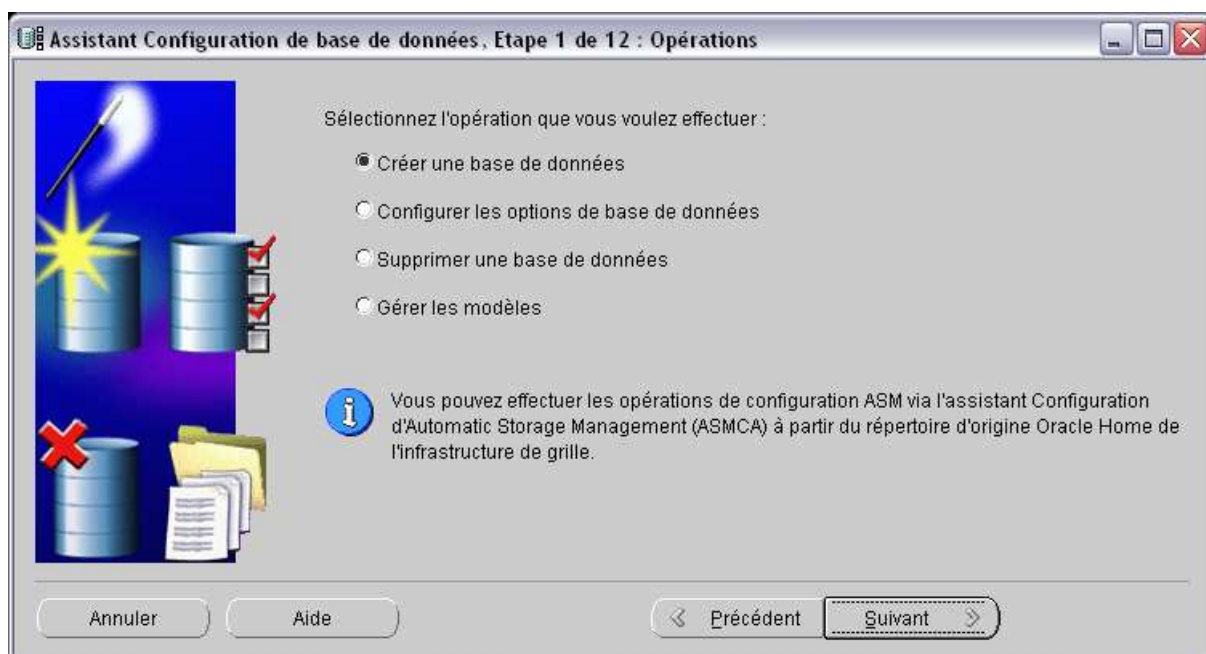
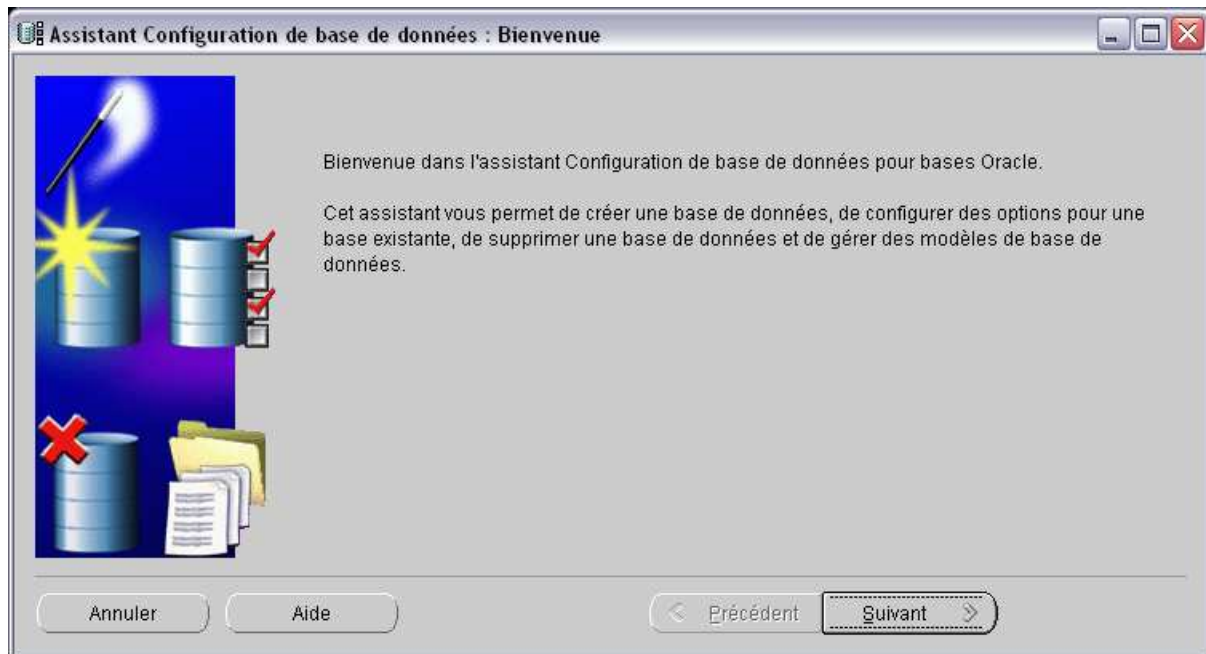
Ces scripts sont détaillés ci-dessous :

```
@C:\app\oracle\admin\tahiti\scripts\JServer.sql
@C:\app\oracle\admin\tahiti\scripts\context.sql
@C:\app\oracle\admin\tahiti\scripts\xdb_protocol.sql
@C:\app\oracle\admin\tahiti\scripts\ordinst.sql
@C:\app\oracle\admin\tahiti\scripts\interMedia.sql
@C:\app\oracle\admin\tahiti\scripts\cwmlite.sql
@C:\app\oracle\admin\tahiti\scripts\labelSecurity.sql
@C:\app\oracle\admin\tahiti\scripts\emRepository.sql
@C:\app\oracle\admin\tahiti\scripts\apex.sql
@C:\app\oracle\admin\tahiti\scripts\owb.sql
@C:\app\oracle\admin\tahiti\scripts\netExtensions.sql
@C:\app\oracle\admin\tahiti\scripts\lockAccount.sql
@C:\app\oracle\admin\tahiti\scripts\postDBCreation.sql
```





12.2 Présentation de l'outil DBCA



Assistant Configuration de base de données, Etape 2 de 12 : Modèles de base de données

Modèles avec fichiers de données contenant des bases de données précréées. Ils vous permettent de créer une base de données en quelques minutes, plutôt qu'en une heure ou plus. N'utilisez les modèles sans fichiers de données que si nécessaire, tel que lorsque vous devez modifier des attributs comme la taille de bloc, qui ne peut pas être modifiée après la création de la base de données.

Sélectionner	Modèle	Inclut les fichiers de données
<input type="radio"/>	BD généraliste ou traitement transactionnel	Oui
<input checked="" type="radio"/>	Base de données personnalisée	Non
<input type="radio"/>	Data Warehouse	Oui

[Afficher les détails...](#)

Annuler Aide Précédent Suivant

Assistant Configuration de base de données, Etape 5 de 12 : informations d'identification et de connexion de la base de données

Pour des raisons de sécurité, vous devez indiquer des mots de passe pour les comptes utilisateur suivants dans la nouvelle base de données.

☐ Utiliser des mots de passe d'administration différents

Nom utilisateur	Mot de passe	Confirmer le mot de passe
SYS		
SYSTEM		
DBSNMP		
SYSMAN		

☒ Utiliser le même mot de passe d'administration pour tous les comptes

Mot de passe :

Confirmez le mot de passe :

Annuler Aide Précédent Suivant



ATTENTION, à partir de la version 11g, les mots de passe sont sensibles à la casse.



Assistant Configuration de base de données, Etape 9 de 11 : Paramètres d'initialisation

Mémoire Dimensionnement Jeux de caractères Mode de connexion

☒ Standard

Taille de la mémoire (SGA et PGA) : 300 MB

Pourcentage : 30 % 300 MB 1015 MB

☒ Utiliser la gestion automatique de la mémoire Afficher la répartition de la mémoire...

☐ Personnalisé

Gestion de la mémoire : Gestion automatique de la mémoire partagée

Taille de la mémoire SGA : 304 Mégaoctets

Taille de la mémoire PGA : 101 Mégaoctets

Mémoire totale pour Oracle : 406 Mégaoctets

Tous les paramètres d'initialisation...

Annuler Aide Précédent Suivant Terminer

Assistant Configuration de base de données, Etape 10 de 11 : Stockage de base de données

Stockage

- Fichier de contrôle
- Espaces disque logiques
 - SYSAUX
 - SYSTEM
 - TEMP
 - UNDOTBS1
 - USERS
- Fichiers de données
 - {ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}\sysaux01.dbf
 - {ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}\system01.dbf
 - {ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}\temp01.dbf
 - {ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}\undotbs01.dbf
 - {ORACLE_BASE}\oradata\{DB_UNIQUE_NAME}\users01.dbf
- Groupes de fichiers de journalisation
 - 1
 - 2
 - 3

Nom	Type	Gestion des ensembles de blocs contigus
SYSAUX	PERMANENT	LOCAL
SYSTEM	PERMANENT	LOCAL
TEMP	TEMPORARY	LOCAL
UNDOTBS1	UNDO	LOCAL
USERS	PERMANENT	LOCAL

Créer Supprimer Variables d'emplacement de fichier...

Annuler Aide Précédent Suivant Terminer



12.3 Valeurs des paramètres

Les paramètres utilisés dans le fichier `SPFILE` sont modifiables. Ils représentent les valeurs suivantes :

⇒ **DB_NAME**

Nom de la base (jusqu'à 8 caractères)

Généralement égal au nom de l'instance (`ORACLE_SID`)

⇒ **DB_DOMAIN**

Localisation logique de la base sur le réseau (jusqu'à 128 caractères)

Permet à Oracle de construire le nom global de la base = `DB_NAME.DB_DOMAIN`

Important si la base appartient à un système distribué (ou est susceptible de l'être)

Valeur par défaut : `WORLD`

`DB_DOMAIN = PARIS.ORA.FR`

⇒ **CONTROL_FILES**

Emplacement des fichiers de contrôle de la base

En spécifier au minimum 2, si possible sur des disques différents (dans l'idéal, un par disque)

`CONTROL_FILES = C:\ORACLE\PRODUCT\10.1.0\ORADATA\TAHITI\CONTROL01.CTL,
D:\ORACLE\PRODUCT\10.1.0\ORADATA\TAHITI\CONTROL02.CTL`

⇒ **NLS_LANGUAGE**

Langage par défaut de l'instance, utilisé pour les messages, la date et l'heure. La valeur par défaut est dérivée du paramètre `NLS_LANG`.

`NLS_LANGUAGE = french`

⇒ **NLS_TERRITORY**

Territoire par défaut de l'instance, utilisé pour la numérotation des jours et des semaines. Détermine également la valeur par défaut des formats de date, des séparateurs numériques et des symboles monétaires.

`NLS_TERRITORY = France`

⇒ **DB_BLOCK_SIZE**

Taille en octets d'un bloc de données (compris entre 2 ko et 32 ko)

Doit être un multiple de la taille de bloc du système d'exploitation

Ne peut pas être modifié ultérieurement sans recréer la base

`DB_BLOCK_SIZE = 8192`

⇒ **COMPATIBLE**

Paramètre de compatibilité, prend la valeur 11.2.0.0 par défaut.

`compatible = 11.2.0.0`

⇒ **DIAGNOSTIC_DEST**

Apparu en version 11, ce paramètre définit la destination des fichiers de trace générés par la base de données.

`diagnostic_dest='C:\app\oracle'`



⇒ **MEMORY_TARGET**

Apparu en version 11, si ce paramètre a une valeur différente de zéro, la gestion automatique de la mémoire est activée. Dans ce cas les paramètres `SGA_TARGET` et `PGA_AGGREGATE` sont dépréciés. Oracle aura une réserve de mémoire vive en cas de besoin.

`memory_target=425721856`

⇒ **MEMORY_MAX_SIZE**

Apparu en version 11, ce paramètre délimite la taille totale de la SGA et de la PGA utilisée par l'instance sur le serveur. Il doit être adapté à `MEMORY_TARGET`.

`memory_max_size=425721856`

⇒ **REMOTE_LOGIN_PASSWORDFILE**

A positionner selon la stratégie adoptée pour l'identification `SYSDBA`

`NONE` = pas de fichier de mots de passe – identification par l'OS

`EXCLUSIVE` = utilisation d'un fichier de mots de passe dédié à une base

`SHARED` = utilisation d'un fichier de mots de passe partagé entre plusieurs bases

`REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE`

⇒ **UNDO_TABLESPACE**

Permet de spécifier le nom du tablespace contenant les segments d'annulation.

Si le nom du tablespace spécifié ne correspond pas au nom du tablespace `UNDO` de la base une erreur apparaîtra dans le fichier des alertes.

Valeur par défaut : chaque base de données contient 0 ou plusieurs espaces disque logiques d'annulation. En mode `SMU`, un seul espace disque logique de ce type est affecté à chaque instance `ORACLE`.

`UNDO_TABLESPACE = UNDOTBS`

⇒ **PROCESSES**

Permet de limiter le nombre de processus simultanés sur le serveur.

Pour connaître le nombre de processus d'arrière plan utilisez la vue `V$BGPROCESS`.

⇒ **OPEN_CURSOR**

Nombre maximum de curseurs ouverts en simultané. Compter 1 pour chaque session ouverte en simultanée et un pour chaque utilisateur interne à Oracle comme `SYSMAN` ou `DBSNMP`.

Ouvrir un grand nombre de curseurs évite une erreur de dépassement et n'a aucune incidence sur la base.

`OPEN_CURSOR = 500`

⇒ **CURSOR_SHARING = EXACT**

Description : ce paramètre contrôle les instructions SQL qui peuvent partager le même curseur.

Plage de valeurs :

`FORCE` : oblige les instructions ne différant que par certains littéraux à partager un curseur, à moins que les littéraux ne modifient le sens de l'instruction.

`EXACT` : seules les instructions SQL identiques partagent un curseur.

Valeur par défaut : `EXACT`



⇒ **STATISTICS_LEVEL**

Niveau de collecte des statistiques sur la base de données et le système utilisés.

Valeurs possibles : BASIC, TYPICAL (par défaut), ALL

BASIC désactive la gestion automatique des statistiques

TYPICAL permet de bénéficier des fonctionnalités de la gestion automatique de la version 10g

ALL collecte d'avantage de statistiques mais a un impact sur les performances

⇒ **CLUSTER_DATABASE_INSTANCES = 1**

Description : nombre d'instances actuellement configurées comme éléments de la base de données de cluster. Ce paramètre permet de définir la taille des structures SGA, qui dépend du nombre d'instances configurées. L'attribution d'une valeur appropriée à ce paramètre optimisera l'utilisation de la mémoire SGA. Plusieurs paramètres sont calculés via ce nombre.

Plage de valeurs : toute valeur non nulle

Valeur par défaut : 1

⇒ **CLUSTER_DATABASE = FALSE**

Description : paramétrer CLUSTER_DATABASE sur TRUE pour activer l'option Real Application Clusters.

Plage de valeurs : TRUE | FALSE

Valeur par défaut : FALSE

⇒ **DB_RECOVERY_FILE_DEST**

Emplacement de la zone de récupération rapide (*flash recovery area*). Si ce paramètre est spécifié, il faut spécifier le paramètre DB_RECOVERY_FILE_DEST_SIZE.

DB_RECOVERY_FILE_DEST = d:\oracle\Flash_recovery_area

⇒ **DB_RECOVERY_FILE_DEST_SIZE**

Taille maximum autorisée des fichiers stockés dans la zone de récupération rapide, définie en octets, Ko (K), Mo (M) ou en Go (G).

DB_RECOVERY_FILE_DEST_SIZE = 30G

⇒ **AUDIT_FILE_DEST = {ORACLE_BASE}\ADMIN\{DB_UNIQUE_NAME}\ADUMP**

Description : chaque connexion SYSDBA ou INTERNAL à la base de données génère un fichier d'audit dans ce répertoire (UNIX uniquement).

Plage de valeurs : tout nom de répertoire valide

Valeur par défaut : ORACLE_HOME/rdbms/audit

⇒ **AUDIT_TRAIL = DB**

Description : active ou désactive l'option d'audit de la base de données. Les enregistrements d'audit sont écrits dans la table SYS.AUD\$ lorsque le paramètre a la valeur TRUE ou DB, ou dans un fichier du système d'exploitation lorsque le paramètre a la valeur OS.

Plage de valeurs : NONE | FALSE | DB | TRUE | OS

Valeur par défaut : NONE

⇒ **CORE_DUMP_DEST = ?\RDBMS\TRACE**

Description : nom de répertoire, indiquant l'emplacement de vidage de la mémoire (sous UNIX). Plage de valeurs : tout nom de répertoire valide

Valeur par défaut : ORACLE_HOME/dbs



12.4 Vues du dictionnaire de données

Les vues du dictionnaire de données intéressantes sont :

- V\$INSTANCE : informations sur l'instance
- V\$DATABASE : informations sur la base de données
- V\$VERSION : informations sur la version Oracle utilisée par la base de données
- DATABASE_PROPERTIES : informations sur les propriétés par défaut de la base de données

12.5 EMCA : Création de l'OEM repository (Database Control)

La plupart du temps l'installation d'entreprise manager se fait lors de la création de la base (si vous n'optez pas pour le grid control), mais vous pouvez toujours laisser l'installation d'entreprise manager après la création de la base.

Pour l'installer il y a plusieurs méthodes soit l'outil graphique DBCA, soit un programme en ligne de commande EMCA (*Enterprise Manager Configuration Assistant*) c'est celui dont nous allons parler. la syntaxe générale de l'outil EMCA est :

```
Emca operation mode flag parameters
```

Vous pouvez voir la liste complète en tapant :

```
>emca -h
```

Pour créer votre console d'administration la création de la base tapez :

```
$ emca -config dbcontrol db -repos create

STARTED EMCA at jun 06, 2010 9:21:39 PM
EM Configuration Assistant, Version 10.2.0.1.0 Production
Copyright © 2003, 2005, Oracle. All rights reserved.

Enter the following information:
Database SID: db10
Listener port number: 1521
Password for SYS user: change on install
Password for DBSNMP user: manager
Password for SYSMAN user: manager
Email address for notifications (optional):
Outgoing Mail (SMTP) server for notifications (optional):
-----

You have specified the following settings

Database ORACLE HOME ..... /u01/app/oracle/product/10.2.0/db 1
Database hostname ..... dbserver
Listener port number ..... 1521
Database SID ..... db10
Email address for notifications .....
Outgoing Mail (SMTP) server for notifications .....
-----
```



```
Do you wish to continue? [yes(Y)/no(N)]: Y
jun 06, 2010 10:00:12 PM oracle.sysman.emcp.EMConfig perform
INFO: This operation is being logged at
/u01/app/oracle/product/10.2.0/db_1/cfgtoollogs/emca/db10/emca_2010-01-06_09-21-
39-PM.log.
jun 06, 2010 10:00:15 PM oracle.sysman.emcp.EMReposConfig createRepository
INFO: Creating the EM repository (this may take a while) ...
jun 06, 2010 10:05:51 PM oracle.sysman.emcp.EMReposConfig invoke
INFO: Repository successfully created
jun 06, 2010 10:06:01 PM oracle.sysman.emcp.util.DBControlUtil startOMS
INFO: Starting Database Control (this may take a while) ...
jun 06, 2010 10:07:49 PM oracle.sysman.emcp.EMDBPostConfig performConfiguration
INFO: Database Control started successfully
jun 06, 2010 10:07:49 PM oracle.sysman.emcp.EMDBPostConfig performConfiguration
INFO: >>>>>>>> The Database Control URL is http://dbserver:1158/em <<<<<<<<<
Enterprise Manager configuration completed successfully
FINISHED EMCA at jun 06, 2010 10:07:49 PM
```

Vous devez fournir quelques informations comme le SID le port listener., cela prends quelques minutes et vous pouvez suivre les différentes étapes de création,

Une fois le programme terminé noter l'adresse URL qui apparait :

⇒ `http(s)://nommachine:port/em` est lancer la avec votre navigateur

Pour recréer la console il suffit de changer l'ordre CREATE par RECREATE dans la commande précédente.

Si vous êtes voulez désinstaller la DB Console, utilisez la commande suivante:

```
$ emca -deconfig dbcontrol db -repos drop

STARTED EMCA at jun 06, 2006 9:53:55 PM
EM Configuration Assistant, Version 10.2.0.1.0 Production
Copyright © 2003, 2005, Oracle. All rights reserved.

Enter the following information:
Database SID: db10
Listener port number: 1521
Password for SYS user: change on install
Password for SYSMAN user: manager

Do you wish to continue? [yes(Y)/no(N)]: Y
jun 06, 2006 9:54:15 PM oracle.sysman.emcp.EMConfig perform
INFO: This operation is being logged at
/u01/app/oracle/product/10.2.0/db_1/cfgtoollogs/emca/db10/emca_2006-01-06_09-53-
55-PM.log.
jun 06, 2006 9:54:16 PM oracle.sysman.emcp.util.DBControlUtil stopOMS
INFO: Stopping Database Control (this may take a while) ...
jun 06, 2006 9:54:35 PM oracle.sysman.emcp.EMReposConfig dropRepository
INFO: Dropping the EM repository (this may take a while) ...
jun 06, 2006 9:56:48 PM oracle.sysman.emcp.EMReposConfig invoke
INFO: Repository successfully dropped
Enterprise Manager configuration completed successfully
FINISHED EMCA at jun 06, 2006 9:56:48 PM
```



Pour vérifier le statut de la DB Console :

```
|>emctl status dbconsole—qui fournit le statut de la console web.
```

Et :

```
|>emctl status agent—pour le status de l'agent d'entreprise manager
```

Vous pouvez arrêter la console avec :

```
|>emctl stop dbconsole
```

Ou la démarrer avec :

```
|>emctl start dbconsole
```



13 Automatiser le démarrage de la base

Automatiser le démarrage et l'arrêt de la base lors du démarrage ou de l'arrêt du système dépend de la plate-forme.

13.1 Sous unix

Dans le fichier `/etc/oratab`, mettre une entrée pour chaque instance avec le format suivant :

```
|<ORACLE_SID>:<ORACLE_HOME>:{Y|N}
```

```
|TAHITI:/u01/app/oracle/product/10.1.0.3.:Y
```

Au démarrage et à l'arrêt, le système appelle les scripts `dbstart` et `dbshut` qui lisent le fichier `oratab` pour identifier les bases à démarrer ou arrêter, ces scripts peuvent éventuellement être appelés manuellement pour démarrer ou arrêter les bases configurées à « Y » dans `oratab`.

13.2 Sous Windows

Pour démarrer automatiquement une base au démarrage du système, il faut :

- ♦ Mettre le service (OracleService<SID>) associé à l'instance en démarrage automatique
- ♦ S'assurer que dans la base de registre (HKEY_LOCAL_MACHINE\ SOFTWARE\ORACLE\HOME_x) ,
ORA_<SID>_AUTOSTART est à TRUE
- ♦ ORA_<SID>_PFILE chemin + nom du fichier de paramètres texte standard, vide ou inexistant pour un fichier de paramètres serveur. Pour démarrer avec un autre fichier de paramètres serveur, utilisez la technique du fichier de paramètres texte contenant un paramètre `SPFILE`

Problèmes liés au fichier de paramètres serveur `SPFILE` :

- ♦ Si le paramètre ORA_<SID>_PFILE contient une valeur erronée, l'instance ne redémarre pas.
- ♦ Si le paramètre ORA_<SID>_PFILE est vide ou n'existe pas, la séquence de recherche d'un fichier de paramètres texte ou serveur s'effectue en suivant la séquence du startup.
 - ⇒ spfile<SID>.ora
 - ⇒ spfile.ora (!)
 - ⇒ init<SID>.ora

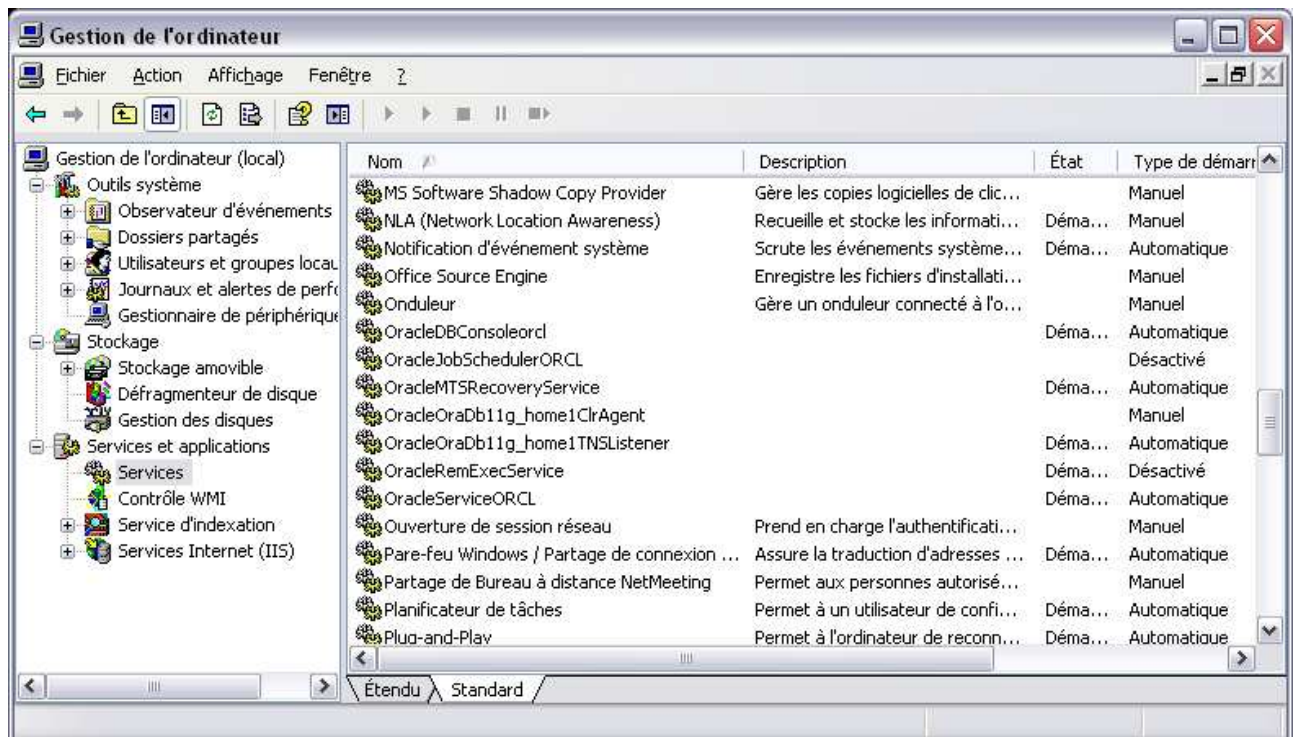
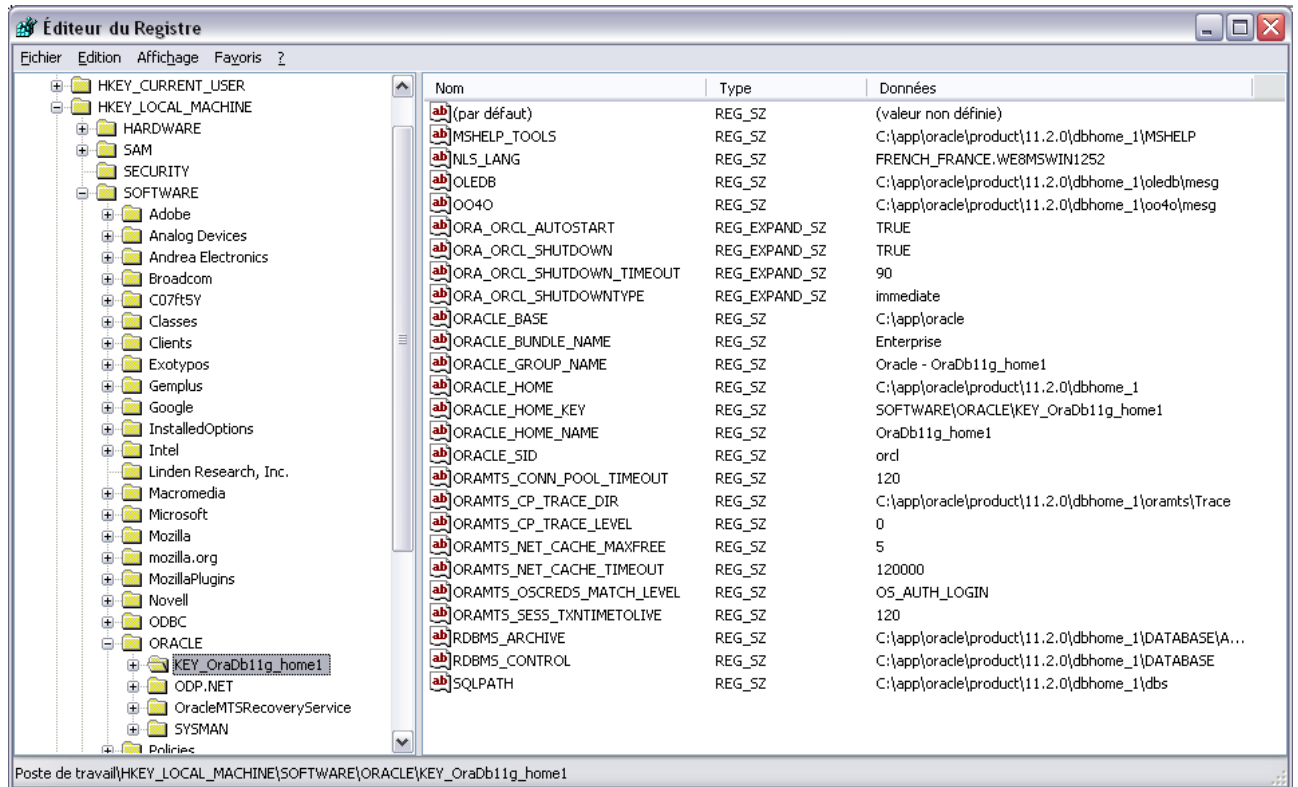
Pour arrêter automatiquement une base lors de l'arrêt du système, il faut :

- ♦ S'assurer que dans la base de registre :

HKEY_LOCAL_MACHINE\ SOFTWARE\ORACLE\HOME_x, ORA_<SID>_SHUTDOWN est à TRUE

et ajuster éventuellement ORA_<SID>_SHUTDOWN_{TYPE} et ORA_<SID>_SHUTDOWN_{TIMEOUT}





14 Accéder à une base distante

Oracle Net permet à des produits Oracle situés sur des machines différentes de communiquer entre eux. Ainsi Oracle Net rend le réseau transparent et permet le transfert de données entre les 2 machines.



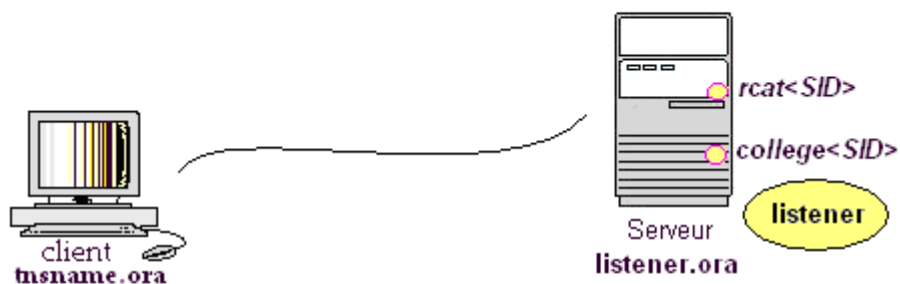
Oracle Net a pour objectif de rendre le réseau « transparent » pour les applications.
- Oracle Net doit être installé sur chaque machine du réseau.

Le processus d'écoute LISTENER placé sur le serveur (coté instance de base de données) permet la connexion des clients à l'instance. Celle-ci est vue comme un service (paramètre SERVICES_NAME dans l'instance). Le LISTENER est configuré via le fichier LISTENER.ORA.

Plusieurs méthodes de connexions peuvent être utilisées :

- ◆ Locale : le fichier TNSNAMES.ORA est configuré sur le poste client et se charge de la résolution du service Oracle Net.
- ◆ Simplifiée, (easy connect naming), permettant une connexion via l'adresse du service à travers le réseau TCP/IP.
- ◆ LDAP (directory naming), un annuaire LDAP se charge de la résolution du nom de service. Cette méthode nécessite un produit tiers.

A l'adresse indiquée le listener reçoit la demande et connecte le client à l'instance <SID> demandée



Le client se connecte en utilisant le nom du service Oracle Net

connect USER01/motdepasse@Service

(le tnsname.ora associe le nom du service à l'adresse du serveur)

Fonctionnement de Oracle Net



14.1 Configuration coté serveur

Pour permettre à un client de se connecter à une base de données distante, il faut d'abord configurer le LISTENER.

Le LISTENER se matérialise par un service (Oracle<NomHome>TNSListener) sur plate-forme Windows ou par un processus (tnslsnr) sur plate-forme Unix.

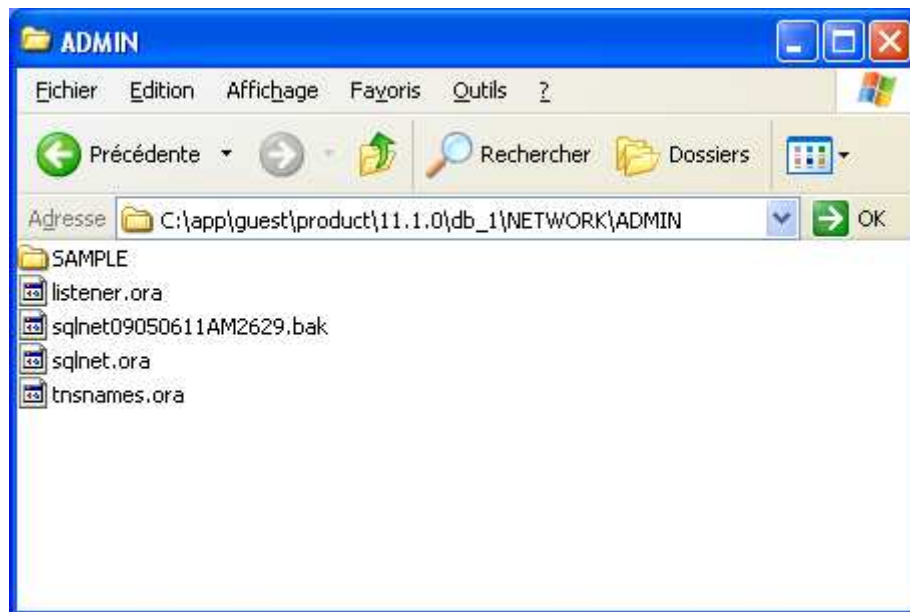
Il est configuré par le fichier listener.ora.

Les instances enregistrent automatiquement leur(s) service(s) auprès du processus d'écoute. Celui-ci est déclaré grâce au paramètre SERVICE_NAMES.

Le listener peut écouter à plusieurs emplacements (protocoles différents, ou variantes du même protocole par exemple 2 ports en TCP/IP), et peut écouter pour plusieurs bases de données et éventuellement pour des versions d'Oracle différentes.

Pour que des postes distants puissent se connecter à la base il faut que le LISTENER soit lancé.

Le LISTENER est utilisé à la première connexion d'un utilisateur. Après la connexion, un arrêt et un redémarrage du LISTENER ne déconnecte pas les utilisateurs déjà connectés.



Le LISTENER s'administre grâce à l'outil LSNRCTL :

```
C:\>LSNRCTL
LSNRCTL for 32-bit Windows: Version 10.2.0.1.0 - Production on 13-AO•T -2006 02:
26:41
Copyright © 1991, 2005, Oracle. All rights reserved.
Bienvenue Ó LSNRCTL, tapez « aide » pour plus d'informations.

LSNRCTL>
```



LSNRCTL permet notamment d'arrêter et de démarrer le LISTENER.

En cas de problème de connexion à partir d'un poste client, vérifier que le LISTENER est bien lancé (ne pas hésiter à le redémarrer).

La configuration côté serveur consiste à configurer le LISTENER, c'est à dire indiquer comment et pour quelles bases il écoute.

Cette configuration peut se faire en modifiant directement le fichier Listener.ora ou en utilisant l'assistant Oracle Net.

Listener.ora

```
# listener.ora Network Configuration File:
C:\oracle\product\10.2.0\db_1\network\admin\listener.ora
# Generated by Oracle configuration tools.
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = C:\oracle\product\10.2.0\db_1)
      (PROGRAM = extproc)
    )
  )
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1))
      (ADDRESS = (PROTOCOL = TCP) (HOST = TELLORA01) (PORT = 1521))
    )
  )
```

Les principales commandes rattachées au LISTENER « lsnrctl » sont les suivantes :

- ◆ Start = démarrer le listener
- ◆ Stop = arrêter le listener
- ◆ Status = obtenir le statut du listener
- ◆ Reload = réinitialiser le listener
- ◆ Exit = sortir de lsnrctl
- ◆ Save_config = crée une sauvegarde du fichier listener.ora puis met à jour le fichier avec les paramètres modifiés à l'aide de lsnrctl.
- ◆ Services = affiche les services disponibles ainsi que l'historique de connexion.
- ◆ Help = affiche une liste d'options de commande de l'utilitaire lsnrctl.
- ◆ Quit = quitter l'utilitaire et revenir à l'invite du système d'exploitation.
- ◆ Version = affiche des informations de version sur le listener.
- ◆ Show = affiche les valeurs courantes des paramètres.
- ◆ Set password mot_de_passe = permet de se connecter au listener via un mot de passe.



Pour vérifier l'exécution du LISTENER, sous le système d'exploitation exécuter la commande :

```
Microsoft Windows XP [version 5.1.2600]
© Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrateur>lsnrctl
LSNRCTL for 32-bit Windows: Version 10.1.0.3.0 - Production on 23-MARS -2005 10:48:46
Copyright © 1991, 2004, Oracle. All rights reserved.
Welcome to LSNRCTL, type "help" for information.

LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC) (KEY=EXTPROC)))
STATUS of the LISTENER
Alias                LISTENER
Version              TNSLSNR for 32-bit Windows: Version 10.1.0.3.0 - Production
Start Date           23-MARS -2005 10:48:26
Uptime                0 days 0 hr. 0 min. 24 sec
Trace Level           off
Security              ON: Local OS Authentication
SNMP                 OFF
Listener Parameter File D:\oracle\product\10.1.0\Db_1\network\admin\listener.ora
Listener Log File     D:\oracle\product\10.1.0\Db_1\network\log\listener.log
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (PIPENAME=\\.\pipe\EXTPROCipc)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=ATTOUCHE) (PORT=1521)))
Services Summary...
Service "orcl" has 1 instance(s).
Instance "orcl", status UNKNOWN, has 1 handler(s) for this service...
The command completed successfully
```

14.2 Configuration coté client

La configuration coté client se fait en modifiant le fichier tnsname.ora, et en lui ajoutant l'accès à la nouvelle instance.

Il se trouve dans le répertoire :

➡ **D:\oracle\product\10.2.0\db_1\NETWORK\ADMIN**

```
# tnsnames.ora Network Configuration File:
C:\oracle\product\10.2.0\db_1\network\admin\tnsnames.ora
# Generated by Oracle configuration tools.

TAHITI =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP) (HOST = TELLORA01) (PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = tahiti)
    )
  )

EXTPROC_CONNECTION_DATA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC1))
    )
    (CONNECT_DATA =
      (SID = PLSExtProc)
      (PRESENTATION = RO)
    )
  )
```





Le fichier tnsname.ora ne contient aucune information relative au poste client, il est donc possible d'en créer un et de le diffuser sur d'autres postes.

14.3 Changer de machine automatiquement

Si vous avez installé une base de données de secours, vous pouvez autoriser le changement de machine de façon automatique. Pour cela, mettre dans le Tnsnames l'adresse de la nouvelle machine ou est monté le Data Guard.

En cas de panne, la bascule se fait sans que l'utilisateur s'en aperçoive, et sans intervention de l'administrateur au niveau du TNSNAMES.ORA, par contre il faudra que l'administrateur passe le Data Guard en base primaire grâce à un Switchover.

```
TAHITI =
(DESCRIPTION =
  (LOAD_BALANCE = OFF)                pour le RAC
  (FAILOVER = ON)
  (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP) (HOST = poste01) (PORT = 1521))
    (ADDRESS = (PROTOCOL = TCP) (HOST = poste02) (PORT = 1521))  pour le Data Guard
  )
  (CONNECT_DATA =
    (SERVER = DEDICATED)
    (SERVICE_NAME = tahiti)
  )
)
```

14.4 EZCONNECT

En version Oracle 10g vous pouvez vous connecter sans tnsname.ora grâce à ezconnect :

Il suffit de configurer le fichier sqlnet.ora.

➡ C:\oracle\product\10.2.0\db_1\NETWORK\ADMIN\sqlnet.ora

```
# sqlnet.ora Network Configuration File:
C:\oracle\product\10.2.0\db_1\network\admin\sqlnet.ora
# Generated by Oracle configuration tools.
# This file is actually generated by netca. But if customers choose to
# install "Software Only", this file wont exist and without the native
# authentication, they will not be able to connect to the database on
NT.

SQLNET.AUTHENTICATION_SERVICES= (NTS)
NAMES DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
```




```
connect USER/MotPasse@[host]:[port]/[service name]
```

Se connecter en tapant la commande :

```
Sqlplus /nolog
• connexion traditionnelle via le tnsnames
SQL> connect system/tahiti@tahiti
Connecté.
SQL>

C:\Documents and Settings\ATTOUCHE Clotilde>hostname
TELLORA01

• dans SQLPLUS
• connexion via ezconnect
SQL> connect system/tahiti@//tellora01:1521/tahiti
Connecté.
```

14.5 Bases distantes et database Links

Le Database Link est un lien qui permet l'accès à des objets situés dans une base de données distante.

Il est configuré à partir d'Oracle Net et correspond au « service » oracle Net défini dans le listener.

```
create [ public ] database link nom_lien
[ connect to nom user identified by mot_passe ]
using chaine_de_connection
```

Le nom de l'utilisateur de connexion précisé doit être déclaré dans la base de données distante.

La chaîne de connexion est déclarée dans le « listener.ora » sur une base pour permettre l'accès de celle-ci à la base distante, et dans le fichier « tname.ora ».

Le fichier listener.ora permet aux postes client d'accéder au serveur Oracle via Oracle Net.

Le fichier tname.ora permet au serveur Oracle d'accéder à une autre machine via Oracle Net.

Exemples

```
create public database link dli_classe
using 'calan_tcp_LYCE';
drop database link dli_classe ;
create database link dli_clastest.world
connect to admindba identified by oracle
using 'caladan_tcp_LYCE';
```



Le tsname.ora

```
caladan_tcp_LYCE=
(DESCRIPTION=
(ADDRESS=
(PROTOCOL=TCP)
(HOST=10.150.160.106)
(PORT=1521))
(CONNECT_DATA=(SID=LYCE))
```

Utilisation d'un DB LINK

Exemples d'utilisation d'un DB_Link pour faire des insertions sur un site distant.

```
insert into clo01.classe@dli\_classe
values (97, 'tyty7', 2);
insert into clo01.classe@dli\_classe
values (98, 'tyty8t', 2);
```



15 Sécuriser la base de données

Assurer la sécurité des données est une des tâches principales de l'administrateur.

Cette sécurité est assurée par la mise en œuvre d'une protection des fichiers sensibles de la base de données :

- ⇒ Fichiers de contrôle
- ⇒ Fichiers de redo log

15.1 Le fichier de contrôle

Lorsqu'une instance est lancée pour ouvrir une base de données, le fichier de contrôle est le premier fichier ouvert, il permet ensuite à l'instance de localiser et d'ouvrir les autres fichiers de la base de données. Si on perd le fichier de control, la base de données reste à l'état NOMOUNT, et ne pourra pas s'ouvrir.

Le fichier de contrôle est automatiquement mis à jour par Oracle lors de chaque modification de la structure de la base de données (ajout ou déplacement de fichier, ...).

Un fichier de contrôle contient les informations suivantes :

- Le nom et l'identifiant de la base de données
- Le nom et l'emplacement des fichiers de données et des fichiers Redo Log
- Le nom des tablespaces
- La date et l'heure de création de la base de données
- Le numéro de séquence du journal courant
- Des informations relatives au point de synchronisation
- L'historique du journal
- Les informations de sauvegarde de l'utilitaire Recovery Manager

Lors de sauvegardes faites avec l'utilitaire Recovery Manager (RMAN), certaines vues du dictionnaire et certaines commandes RMAN permettent d'interroger le fichier de contrôle pour connaître l'état des sauvegardes réalisées.

Quand RMAN sauvegarde une base de données (appelée base de données cible) des informations sont toujours consignées dans le fichier de contrôle de la base cible. Pour limiter la taille des fichiers de contrôle, les anciennes entrées de sauvegarde sont écrasées après un certain nombre de jour. Ce nombre de jours limité est défini par le paramètre `CONTROL_FILE_RECORD_KEEP_TIME`. Par défaut ce nombre de jour est égal à 7, soit une semaine. Il doit être augmenté lorsque RMAN utilise le fichier de contrôle de la base cible (15 jours).

Si le fichier de contrôle est trop petit il faudra le recréer en utilisant une trace du fichier de contrôle.

Les fichiers de contrôle doivent être sauvegardés lors de chaque sauvegarde complète ou partielle de la base de données, de plus entre deux sauvegardes « normales », il est conseillé de sauvegarder le fichier de contrôle après toute restructuration importante de la base (ajout/déplacement de fichiers de données ou de redo log).



Pour effectuer une sauvegarde du fichier de contrôle, utiliser l'ordre SQL : ALTER DATABASE

```
ALTER DATABASE BACKUP CONTROLFILE TO  
to [ nouveau nom | trace ] [ reuse ]  
;
```

- Le paramètre TRACE génère un script dans les fichiers traces utilisateur (diagnostic_dest) .
- Cette trace peut être intéressante dans certaines situations.

Trace du fichier de control
optimum_ora_1396.trc

```
*** 2004-06-14 10:46:04.000  
# The following are current System-scope REDO Log Archival related  
# parameters and can be included in the database initialization file.  
#  
# LOG_ARCHIVE_DEST= »  
# LOG_ARCHIVE_DUPLEX_DEST= »  
#  
# LOG_ARCHIVE_FORMAT=ARC%S.%T  
# REMOTE_ARCHIVE_ENABLE=TRUE  
# LOG_ARCHIVE_MAX_PROCESSES=2  
# STANDBY_FILE_MANAGEMENT=MANUAL  
# STANDBY_ARCHIVE_DEST=%ORACLE_HOME%\RDBMS  
# FAL_CLIENT= »  
# FAL_SERVER= »  
#  
# LOG_ARCHIVE_DEST_1='LOCATION=D:\oracle9\ora92\RDBMS'  
# LOG_ARCHIVE_DEST_1='MANDATORY NOREOPEN NODELAY'  
# LOG_ARCHIVE_DEST_1='ARCH NOAFFIRM SYNC'  
# LOG_ARCHIVE_DEST_1='NOREGISTER NOALTERNATE NODEPENDENCY'  
# LOG_ARCHIVE_DEST_1='NOMAX FAILURE NOQUOTA_SIZE NOQUOTA_USED'  
# LOG_ARCHIVE_DEST_STATE_1=ENABLE  
#  
# Below are two sets of SQL statements, each of which creates a new  
# control file and uses it to open the database. The first set opens  
# the database with the NORESETLOGS option and should be used only if  
# the current versions of all online logs are available. The second  
# set opens the database with the RESETLOGS option and should be used  
# if online logs are unavailable.  
# The appropriate set of statements can be copied from the trace into  
# a script file, edited as necessary, and executed when there is a  
# need to re-create the control file.  
#  
# Set #1. NORESETLOGS case  
#  
# The following commands will create a new control file and use it  
# to open the database.  
# Data used by the recovery manager will be lost. Additional logs may  
# be required for media recovery of offline data files. Use this  
# only if the current version of all online logs are available.  
STARTUP NOMOUNT  
CREATE CONTROLFILE REUSE DATABASE "OPTIMUM" NORESETLOGS NOARCHIVELOG  
• SET STANDBY TO MAXIMIZE PERFORMANCE  
MAXLOGFILES 32  
MAXLOGMEMBERS 5  
MAXDATAFILES 128  
MAXINSTANCES 16  
MAXLOGHISTORY 1815  
LOGFILE  
GROUP 1 (  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO01A.LOG',  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO01B.LOG'  
 ) SIZE 10M,  
GROUP 2 (  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO02A.LOG',  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO02B.LOG'  
 ) SIZE 10M,  
GROUP 3 (  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO03A.LOG',  
 'D:\ORACLE9\ORADATA\OPTIMUM\REDO03B.LOG'
```



```
) SIZE 10M
• STANDBY LOGFILE
DATAFILE
'D:\ORACLE9\ORADATA\OPTIMUM\SYSTEM01.DBF',
'D:\ORACLE9\ORADATA\OPTIMUM\UNDOTBS01.DBF',
'D:\ORACLE9\ORADATA\OPTIMUM\ELEVE01.DBF',
'D:\ORACLE9\ORADATA\OPTIMUM\INDX01.DBF',
'D:\ORACLE9\ORADATA\OPTIMUM\OPDEF01.DBF'
CHARACTER SET WE8ISO8859P15
;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# Database can now be opened normally.
ALTER DATABASE OPEN;
# Commands to add tempfiles to temporary tablespaces.
# Online tempfiles have complete space information.
# Other tempfiles may require adjustment.
ALTER TABLESPACE TEMP ADD TEMPFILE 'D:\ORACLE9\ORADATA\OPTIMUM\TEMP01.DBF'
SIZE 10485760 REUSE AUTOEXTEND OFF;
# End of tempfile additions.
#
```



La modification des paramètres définis dans le fichier de contrôle oblige à la recreation de celui-ci !

15.2 Protection du fichier de contrôle

La première action est le multiplexage du fichier de contrôle. Il s'agit de faire une copie du fichier de contrôle qui sera maintenue en miroir par Oracle.



Si la copie du fichier de contrôle n'est pas jugée cohérente par Oracle, une erreur se produira au redémarrage.

Le fichier de contrôle doit être multiplexé pour des raisons de sécurité :

- ⇒ Sans fichier de contrôle, la base ne peut pas démarrer
- ⇒ Au moins deux fichiers de contrôle, si possible sur des disques différents (et dans la pratique, un par disque)



15.2.1 Multiplexer le fichier de contrôle

Faire fonctionner la base avec au moins deux fichiers de contrôle, si possible sur des disques différents (et dans la pratique, un par disque).

Le multiplexage peut être mise en œuvre lors de la création de la base :


- ⇒ Spécifier la liste des fichiers de contrôle souhaités dans le paramètre `CONTROL_FILES` avant d'exécuter l'ordre `SQL CREATE DATABASE`

Mais il peut aussi être mise en œuvre ultérieurement :

- ◆ Arrêter la base proprement (pas `ABORT` !)
- ◆ Dupliquer un fichier de contrôle existant vers une nouvelle destination
- ◆ Mentionner le nouveau fichier de contrôle dans le paramètre `CONTROL_FILES`
- ◆ Redémarrer la base de données

La taille du fichier de contrôle est déterminée par Oracle.

Protéger les fichiers de contrôle est une opération simple. En plus ce sont de petits fichiers, donc on peut en avoir plusieurs copies sans problème.

MODE OPERATOIRE	
	⇒ Exporter le fichier de paramètres serveur <code>CREATE PFILE FROM SPFILE</code>
	⇒ Modifier le fichier <code>SPFILE</code> <code>ALTER SYSTEM ... SCOPE=SPFILE</code>
	⇒ Arrêter la base <code>SHUTDOWN IMMEDIATE</code>
	⇒ Recopier le fichier de contrôle vers le nouvel emplacement
	⇒ Ouvrir la base <code>STARTUP</code>

- Modifier les paramètres `CONTROL_FILES` dans le fichier de paramètres serveur (base ouverte)
`ALTER SYSTEM SET control_files = 'C:\oracle\oradata\tahiti\controlfile01.ctl', 'D:\tahiti\controlfile02.ctl' SCOPE = SPFILE ;/`
- Arrêter la base de données
`Shutdown immediate ;`
- Dupliquer le fichier de contrôle par une commande du système d'exploitation (base fermée)
`HOST COPY C:\oracle\oradata\tahiti\controlfile01.ctl D:\tahiti\controlfile02.ctl`
- Redémarrer la base de données
`Startup`



15.3 Vues du dictionnaire de données

Interroger les vues :

- ⇒ V\$CONTROLFILE
- ⇒ V\$PARAMETER

```
Select name
From v$controlfile ;

NAME
/DISK1/control01.con
/DISK2/control02.con

select value
from v$parameter
where name = 'control_file' ;
```

Pour obtenir des informations sur les différentes sections des fichiers de contrôle, interrogez la vue dynamique sur les performances : V\$CONTROLFILE_RECORD_SECTION

```
Select type, record_size, records_total, records_used
From v$controlfile_record_section
Where type = 'DATAFILE' ;
```

La colonne « record_total » correspond au paramètre MAXDATAFILES de la commande CREATE DATABASE .

15.4 Protection des fichiers de Redo Log

Les fichiers de Redo Log enregistrent toutes les modifications apportées à la base.

Ils sont organisés en groupes composés d'un ou plusieurs membres. Oracle les utilise de manière circulaire, les informations sauvegardées sont donc par défaut périodiquement écrasées. Au minimum la base de données a besoin de 2 groupes.

Ils sont utilisés pour la restauration de la base après un arrêt anormal de celle-ci.

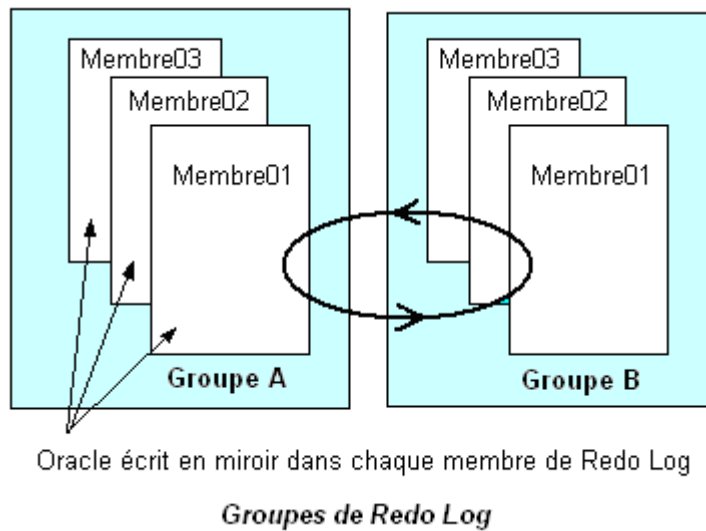
Ils peuvent être réappliqués à une sauvegarde de fichier de données, pour rejouer toutes les modifications survenues entre la sauvegarde et un incident ayant endommagé le fichier (c'est la restauration de média).

LGWR écrit en parallèle dans chaque membre d'un même groupe.

Si un groupe de Redo Logs comporte plusieurs membres et qu'un des membres est indisponible, la base de données peut continuer à fonctionner.

Mettre au minimum deux ou trois membres par groupe.





Quelques possibilités sont apparues avec la version 9i :

- ⇒ Possibilité de forcer l'archivage de façon périodique
Grâce au paramètre ARCHIVE_LAG_TARGET
- ⇒ Possibilité de garantir une durée maximale de restauration d'instance en cas d'arrêt anormal
Grâce au paramètre FAST_START_MTTR_TARGET
- ⇒ Possibilité d'utiliser plusieurs destinations d'archivage
Grâce au paramètre LOG_ARCHIVE_DEST_n, n valant de 1 à 10
Disponible uniquement avec l'édition Entreprise

15.4.1 Dimensionner les fichiers de Redo Log

L'objectif est d'avoir :

- ♦ Un basculement de fichier redo log toutes les 20 à 30 minutes environ.
- ♦ Aucune attente lors d'un basculement de fichier de Redo Log lié à un checkpoint ou à un archivage non terminé.

Dans le fichier des alertes de l'instance surveillez la fréquence des messages

⇒ Relatif au basculement de fichiers redo log

```
Sat Mar 08 07:05:17 2003  
Thread 1 advanced to log sequence 7
```

ET

⇒ Relatif aux attentes lors d'un basculement de fichiers de redo log

```
Sat Mar 08 07:06:48 2003  
Thread 1 cannot allocate new log, sequence 8  
Checkpoint not complete
```

```
Sat Mar 08 07:07:59 2003  
Thread 1 cannot allocate new log, sequence 9  
All online logs needed archiving
```



Recommandations:

En cas de basculement trop rapide il faut augmenter la taille des fichiers de redo log.

En cas d'attente fréquente lors d'un basculement il faut ajouter un groupe de fichiers de redo log.

Si la base est en Archivelog, il faut prévoir un minimum de 3 groupes de Redo Logs.

Si la base possède un DATA Guard prévoir un quatrième groupe de redo Logs, voir un cinquième groupe.

Si la base est montée en RAC (bases en cluster), prévoir 2 groupes de Redo Logs par nœud.

Les opérations d'administration qui peuvent être effectuées sur les fichiers de redo log sont :

- ◆ Ajouter un nouveau groupe de fichiers de redo log permet d'améliorer la disponibilité des fichiers de redo log pour LGWR (en augmentant la durée d'un cycle complet de rotation).
- ◆ Multiplexer les membres de Redo Log pour sécuriser la base de données
- ◆ Déplacer les fichiers de redo log
- ◆ Supprimer un groupe de fichiers de redo log, peut être utilisé dans une opération d'augmentation de la taille des fichiers de redo log (ajout d'un nouveau groupe plus gros puis suppression d'un ancien).
- ◆ Supprimer un membre d'un groupe de fichiers de redo log
- ◆ Forcer le basculement du groupe courant au suivant peut être effectué avant les sauvegardes des archives de Redo Logs.

15.4.2 Multiplexer les fichiers de Redo Log

Le multiplexage des fichiers de redo log peut être mis en œuvre lors de la création de la base de données. Il consiste à dupliquer les membres de Redo Log d'un même groupe.

Il est possible de spécifier plusieurs membres pour chaque groupe listé dans la clause LOGFILE de l'ordre SQL CREATE DATABASE

Il peut aussi être mis en œuvre après création de la base de données, en utilisant de l'ordre SQL ALTER DATABASE.

```
ALTER DATABASE  
ADD LOGFILE MEMBER 'nom_fichier' [,...] TO GROUP numéro  
;
```

```
ALTER DATABASE  
ADD LOGFILE MEMBER 'f:\oracle\oradata\HERMES\redo01.log' TO GROUP 1;
```

Remarques :

- ⇒ La taille du fichier n'a pas besoin d'être spécifiée ; le nouveau fichier a forcément la même taille que les autres membres du groupe
- ⇒ Normalement, tous les groupes doivent avoir le même nombre de membres



15.4.3 Ajouter un groupe de Redo Log

Pour ajouter un nouveau groupe à la base de données utilisez l'ordre SQL ALTER DATABASE :

```
ALTER DATABASE  
ADD LOGFILE [GROUP numéro] spécification_fichier_redo [,...]  
;
```

- spécification_fichier_redo

```
('nom_fichier' [,...]) [ SIZE valeur [K|M] ] [REUSE]
```

Sauf opération d'augmentation de la taille des fichiers de redo log, le nouveau groupe a normalement la même taille que les anciens.

Exemple

```
ALTER DATABASE  
ADD LOGFILE  
GROUP 4 ('d:\oracle\oradata\HERMES\redo04.log',  
'e:\oracle\oradata\HERMES\redo04.log') SIZE 10240K;
```

ATTENTION

On ne peut pas modifier la taille des fichiers de Redo Log, il faut ajouter des groupes ayant une taille souhaitée et supprimer les anciens groupes.

Supprimer un groupe n'est pas possible si c'est le groupe courant. Dans ce cas, la commande permettant de forcer un SWITCH de fichier de Redo Log peut alors être utilisée pour éviter d'attendre. Le SWITCH permettra de changer de groupe.

15.4.4 Déplacer les fichiers de Redo Log


Lorsque l'on déplace un fichier de Redo Log il faut suivre le mode opératoire ci-dessous et veiller à recopier le fichier dans le nouvel emplacement avant d'effectuer la commande RENAME FILE.

Oracle ne s'est pas créer de fichiers ni déplacer des fichiers ou encore créer des répertoires !

Exemple

```
Connect / as sysdba  
Shutdown immediate  
Startup mount  
C:\> host copy e:\oracle\oradata\tahiti\redo04.log  
g:\oracle\oradata\tahiti\redo04.log  
Alter database  
Rename file 'e:\oracle\oradata\tahiti\redo04.log'  
to 'g:\oracle\oradata\tahiti\redo04.log' ;  
Alter database open ;  
Suppression de l'ancien fichier à l'aide d'une commande du système d'exploitation.
```



MODE OPERATOIRE	
	↗ Arrêter la base de données SHUTDOWN IMMEDIATE
	↗ Déplacer les fichier de Redo Log vers le nouvel emplacement COPIER .. +.. COLLER
	↗ Monter la base STARTUP MOUNT
	↗ Indiquer à Oracle le nouvel emplacement ALTER DATABASE RENAME FILE
	↗ Ouvrir la base ALTER DATABASE OPEN

15.4.5 Supprimer un groupe de fichiers redo log

La base doit avoir au moins 3 groupes de fichiers de redo log pour pouvoir en supprimer un (il doit en rester au moins 2).

Le groupe courant (celui dans lequel LGWR est en train d'écrire) ne peut pas être supprimé.

En mode ARCHIVELOG, un groupe pas encore archivé ne peut pas être supprimé.

Les fichiers concernés ne sont pas physiquement supprimés par Oracle, il faudra le faire après en utilisant une commande du système d'exploitation.

Utiliser l'ordre SQL ALTER DATABASE :

```
ALTER DATABASE  
DROP LOGFILE GROUP numéro  
;
```

Exemple

```
ALTER DATABASE  
DROP LOGFILE GROUP 4;
```

15.4.6 Supprimer un membre d'un groupe de redo log

Le groupe concerné doit avoir au moins 2 membres pour pouvoir en supprimer un (il doit en rester au moins un).

Un membre du groupe courant (celui dans lequel LGWR est en train d'écrire) ne peut pas être supprimé.

En mode ARCHIVELOG, un membre d'un groupe pas encore archivé ne peut pas être supprimé.

Les fichiers concernés ne sont pas physiquement supprimés par Oracle.

Pour supprimer tous les membres d'un groupe il faut supprimer le groupe.



Si un membre est corrompu, Oracle lui donne un statut « invalide » dans le dictionnaire de données mais ne le signale pas par un message affiché à l'écran.

Pour détecter qu'un membre est invalide, il faut consulter le fichier des Alertes.

Si un membre d'un groupe est perdu, il faut le supprimer pour mettre à jour le dictionnaire de données.

Utiliser l'ordre SQL ALTER DATABASE :

```
ALTER DATABASE  
DROP LOGFILE MEMBER 'nom fichier' [...]  
;
```

Exemple

```
ALTER DATABASE  
DROP LOGFILE          MEMBER 'e:\oracle\oradata\HERMES\redo01.log' ;
```

15.4.7 Forcer le basculement du groupe courant

Le basculement d'un groupe courant peut être utilisé lorsque l'on a besoin d'effectuer une suppression de groupe ou lorsque l'on veut générer une archive avant d'effectuer une sauvegarde (la base de données doit alors être en ARCHIVELOG).

Utiliser l'ordre SQL ALTER SYSTEM :

```
ALTER SYSTEM SWITCH LOGFILE  
;
```

Si les SWITCH sont trop fréquents, ce n'est pas bon pour les performances.

La vue V\$LOG_HISTORY peut être utilisée pour analyser la fréquence de SWITCH des fichiers de Redo Log (colonne FIRST_TIME).

Le basculement manuel provoque les mêmes événements qu'un basculement automatique

- ♦ Checkpoint : point de reprise
- ♦ Archivage (si l'archivage est activé)

Le paramètre FAST_START_MTTR_TARGET peut être utilisé pour obliger la base à effectuer des points de reprise régulièrement. En effet ce paramètre indique un nombre maximum de secondes pour le redémarrage de l'instance après un arrêt anormal.

Le positionnement de ce paramètre permet à l'instance d'ajuster des points de reprises de manière à pouvoir rejouer l'activité perdue sur la base en respectant les valeurs du paramètre. Attention à ne pas avoir de points de reprises trop fréquents ce qui dégraderait les performances.



15.4.8 Trouver des informations sur les fichiers Redo Log

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les fichiers de redo log :

- ⇒ V\$LOG : informations sur les groupes
- ⇒ V\$LOGFILE : informations sur les membres
- ⇒ V\$LOG_HISTORY : informations sur l'historique des fichiers de redo log
- ⇒ V\$INSTANCE_RECOVERY : informations sur les temps estimés de restauration d'instance.

V\$LOG	
GROUP#	Numéro du groupe
SEQUENCE#	Numéro de séquence du groupe (s'incrémente à chaque SWITCH).
MEMBERS	Nombre de membres
ARCHIVED	Groupe archivé (yes, no)
STATUS	Statut du groupe Unused : groupe jamais utilisé Current : groupe courant Active : groupe encore nécessaire en cas de restauration d'instance (checkpoint non terminé) Inactive : groupe plus nécessaire pour une restauration d'instance (checkpoint terminé)
FIRST_CHANGE#	Plus petit numéro SCN (numéro de transaction) écrit dans le groupe
FIRST_TIME	Date et heure du plus petit numéro SCN

V\$LOGFILE	
GROUP#	Numéro du groupe
STATUS	Statut du groupe Invalide : fichier inaccessible Stale : fichier incomplet (statut des nouveaux membres) Deleted : fichier supprimé Colonne vide : fichier utilisé
MEMBER	Nom complet du fichier membre



\$LOG_ HISTORY	
SEQUENCE#	Numéro de séquence du groupe archivé
FIRST_CHANGE#	Plus petit numéro SCN (numéro de transaction) écrit dans le groupe archivé
NEXT_CHANGE#	Plus grand numéro SCN (numéro de transaction) écrit dans le groupe archivé
FIRST_TIME	Date et heure du plus petit numéro SCN

\$INSTANCE_ RECOVERY	
TARGET_MTTR	Objectif réel de durée de récupération de l'instance recalculée par Oracle.
ESTIMATED_MTTR	Durée de récupération actuelle (tient compte de l'activité de la base)
OPTIMAL_LOGFILE_SIZE	Taille optimale de groupes de Redo Logs en Mo



16 Gestion du stockage

Dans ce chapitre, nous allons faire le lien entre les structures logiques d'Oracle et ces fichiers.

Nous allons étudier comment Oracle stocke les données des tables et des index dans les fichiers et comment ces données sont gérées physiquement sur le disque.

16.1 Notion de tablespace

Un tablespace est une unité logique de stockage composée d'un ou de plusieurs fichiers physiques.

Les fichiers de données sont découpés en blocs d'une taille définie à la création de la base de données (2 ko, 4 ko, 8 ko, etc ...). La taille des blocks Oracle correspondent à un multiple du block géré par le système d'exploitation.

La taille d'un bloc Oracle correspond au paramètre `DB_BLOCK_SIZE`.



L'espace occupé par un objet dans un tablespace est désigné par le terme de segment.

Il y a 4 types de segments gérés dans une base Oracle :

- ◆ Les segments de table = espace occupé par les tables
- ◆ Les segments d'index = espace occupé par les index
- ◆ Les segments d'annulation = espace temporaire utilisé pour stocker les informations permettant d'annuler une transaction (ROLLBACK).
- ◆ Les segments temporaires = espace temporaire utilisé lors d'un tri dans une requête.

La règle est d'utiliser plusieurs tablespaces afin de séparer les différents objets de la base et d'assurer de meilleures performances à la base de données. Cela permettra également d'offrir une plus grande souplesse dans les tâches d'administration.

Chaque type de segment est stocké dans un tablespace qui lui est propre. Ainsi on pourra garantir un minimum de performances.

Ainsi on mettra les tables et les index dans des tablespaces dédiés aux tables ou aux index, les segments temporaires dans des tablespaces temporaires et les segments d'annulation dans le tablespace UNDO.

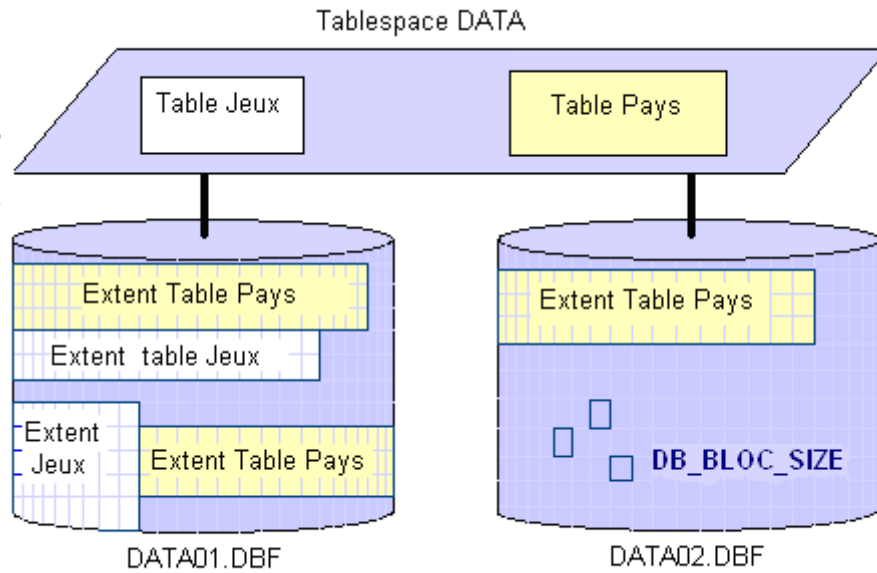
L'espace utilisé par les objets sur le disque est appelé segment (comme les tables par exemple).

A l'intérieur d'un tablespace, le stockage est organisé en segments comportant un ou plusieurs extents (ensemble de bloc Oracle contigus de taille `DB_BLOCK_SIZE` dans un fichier de données).

Lorsqu'un segment est créé dans un tablespace, Oracle lui alloue un premier extent dans un des fichiers du tablespace. Lorsque ce premier extent est plein, Oracle alloue un deuxième extent au segment et ainsi de suite.



Les extents complémentaires alloués au segment sont dans le tablespace d'origine du segment, mais pas forcément côte à côte ni forcément dans le même fichier. Lorsqu'un segment est supprimé, les extents qu'il occupe sont libérés et rendus disponibles pour d'autres segments.



Notion de Segments et d'Extents

Des créations/suppressions fréquentes de données appartenant à un segment ou de segments dans un tablespace conduisent souvent à une fragmentation de l'espace disponible dans ce tablespace.

La clause de stockage permet de gérer la taille du segment à sa création, puis lors de son évolution.

La clause de stockage est créée au moment de la création de l'objet puis peut être modifiée.

Syntaxe :

```
Storage (INITIAL      valeur
         NEXT         valeur
MINEXTENTS valeur
MAXEXTENTS valeur
PCTINCREASE valeur
)
```

- INITIAL taille du premier extent créé à la création de l'objet
- NEXT taille des extents suivants
- MINEXTENTS nombre minimum d'extents créés à la création de l'objet
- MAXEXTENTS nombre maximum d'extents créés pendant la vie de l'objet
- PCTINCREASE pourcentage d'augmentation de la taille d'un extent par rapport au précédent.



La clause de stockage peut être modifiée après création de l'objet !



Exemple

```
-- =====
•      Table : EMPLOYE
-- =====
create table EMPLOYE
(
  ID_EMP      INTEGER          not null,
  NOM         VARCHAR2(30)     not null,
  SALAIRE     NUMBER(4)        not null,
  EMPLOI      VARCHAR2(20)     null ,
  EMP_ID_EMP  INTEGER          null ,
  constraint PK_EMPLOYE primary key (ID_EMP)
  using index
  tablespace INDX
)
Tablespace tbs_local uniform
Storage (initial 400K
next 100K
minextents 2
pctincrease 0
)
/
```



Le tablespace est le plus petit niveau de sauvegarde & restauration.
Le STRIPING consiste à avoir 1 tablespace réparti sur plusieurs disques (le tablespace est alors composé de plusieurs *datafiles*).

Il est conseillé de répartir les différents tablespaces sur des disques différents afin d'éviter les contentions sur les entrées /sorties.

Directives

- ⇒ Ne rien mettre dans les tablespace SYSTEM et SYSAUX.
- ⇒ En plus des tablespaces UNDO, TEMP, et USERS, (penser au suivi des tables et aux sauvegardes car le tablespace est l'unité de stockage) créer :
 - Plusieurs tablespaces pour les tables
 - Plusieurs tablespaces pour les index
- ⇒ En cas d'utilisation de RAID, préférez le RAID 1 (*Mirroring*).
- ⇒ En cas d'utilisation de RAID 5, Beaucoup moins bon pour Oracle, vous pouvez mettre les fichiers de Redo Log sur des disques sans RAID pour éviter les contentions.

16.2 Organisation du stockage dans un tablespace

A partir de la 9i, les tablespaces sont gérés localement et plus par le dictionnaire de données. La gestion des extents est gérée par Oracle et la clause de stockage n'est plus spécifiée à la création du tablespace.

Un des objectifs des tablespaces gérés localement est d'optimiser l'utilisation de l'espace dans les tablespaces et d'éviter le phénomène de fragmentation de l'espace disponible.



Oracle tente d'allouer côte à côte des extents de même taille. Oracle optimise ainsi le stockage et les performances.

```
EXTENT MANAGEMENT  
DICTIONARY | LOCAL [ AUTOALLOCATE | UNIFORM [ SIZE valeur [K|M] ] ]  
[ SEGMENT SPACE MANAGEMENT AUTO ]
```

Attention la taille minimum acceptée par Oracle en UNIFORM SIZE est de 5 blocs.

```
• tablespace géré localement avec des extents uniformes  
CREATE TABLESPACE tbs_uniform  
DATAFILE 'C:\app\guest\oradata\BORA\tbs_uniform.dbf' SIZE 10M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K  
  
• tablespace géré localement avec des extents gérés par Oracle  
CREATE TABLESPACE tbs_auto  
DATAFILE 'C:\app\guest\oradata\BORA\tbs_auto.dbf' SIZE 10M  
EXTENT MANAGEMENT LOCAL AUTOALLOCATE  
SEGMENT SPACE MANAGEMENT AUTO;
```

Les informations relatives à la gestion des espaces (extent libre/alloué) sont enregistrées dans un bitmap, dans l'en-tête de chaque fichier de données :

- ⇒ Chaque bit du bitmap correspond à un extent
- ⇒ Vaut 0 ou 1 selon que l'extent est libre ou alloué



La Bitmap permet d'identifier les extents adjacents libres.

Gestion automatique des extents par Oracle :

- ⇒ 3 blocs pour l'en-tête de fichier plus la taille d'un extent
- ⇒ Taille de l'extent :
 - valeur explicite ou par défaut de la clause SIZE pour un tablespace UNIFORM
 - 64 Ko, 1M, 8M pour un tablespace AUTOALLOCATE

La taille d'extents initialement choisis pour un segment dépend de la taille du segment spécifiée lors de sa création.

- 64Ko pour les segments de moins de 1Mo
- 1Mo pour les segments de moins de 64Mo
- 8Mo pour les segments de moins de 1024Mo



Exemple

Si la table suivante est créée dans un tablespace géré localement avec des extents UNIFORMES de 128 Ko

```
-- =====
•      Table : EMPLOYE
-- =====
create table EMPLOYE
(
  ID_EMP      INTEGER              not null,
  NOM         VARCHAR2(30)         not null,
  SALAIRE     NUMBER(4)            not null,
  EMPLOI      VARCHAR2(20)         null   ,
  EMP_ID_EMP  INTEGER              null    ,
  constraint PK_EMPLOYE primary key (ID_EMP)
  using index
  tablespace INDX
)
Tablespace tbs_local uniform
Storage (initial 400K
next 100K
minextents 2
pctincrease 0
)
/
```

Oracle alloue $(400 + 100) / 128 = 3,5$ arrondi à l'entier supérieur = 4 extents de 128 Ko pour la table.

Si la même table est créée dans un tablespace géré localement avec une gestion automatique des extents Oracle alloue $(400 + 100) / 64 = 7,8$ arrondi à l'entier supérieur = 8 extents de 64 Ko pour la table.



En version 10.2 et en version 11g, la clause Segment Space Management est positionnée à AUTO par défaut.
Elle indique une gestion des segments dont l'en-tête est géré par Bitmap.
-- La Free List disparaît !

16.3 Notion de BIGFILE ou de SMALLFILE

A partir de la version 10g, Oracle introduit la notion de BIGFILE et de SMALLFILE.

La valeur SMALLFILE détermine une gestion de petits fichiers (pouvant contenir au maximum 1022 fichiers d'une taille de 2^{22} blocks = 32Go). alors que la valeur BIGFILE détermine la gestion d'un seul fichier de taille très importante (plus de 4 milliards de blocs) et sont préconisés pour l'utilisation en ASM.

Si aucun type de fichier n'est précisé au moment de la création d'un tablespace, la valeur de la propriété « DEFAULT_TBS_TYPE » est prise par défaut.



```
SQL> -- Afficher les tablespaces définis par défaut
SQL> set pagesize 200
SQL> set linesize 200

SQL> col property_value format A30
SQL> col property_name format A30

SQL> select property_value, property_name
2   from database_properties
3   where property_name like 'DEFAULT%'
4   /

PROPERTY_VALUE                                PROPERTY_NAME
TEMP                                           DEFAULT_TEMP_TABLESPACE
USERS                                         DEFAULT_PERMANENT_TABLESPACE
SMALLFILE                                    DEFAULT_TBS_TYPE
```

La modification de cette option se fait par la commande :

```
ALTER DATABASE SET DEFAULT { SMALLFILE | BIGFILE } TABLESPACE
/
```

La propriété BIGFILE ou SMALLFILE peut être spécifiée à la création du tablespace :

```
CREATE BIGFILE TABLESPACE TBS_GROS
DATAFILE ':\app\oracle\oradata\BORA\DATA_GROS.DBF'
SIZE 20G
/
```

Les recommandations d'oracle vis-à-vis du stockage sont *Strip and Mirror Everything* (SAME).
Soit le RAID 1 = Raid 0 (STRIP) pour les performances + Raid 1 (MIRROR) pour la sécurité.



17 Tablespaces permanents

Les tablespaces permanents sont utilisés pour stocker des objets permanents tels que les tables ou les index. C'est l'unité logique de stockage d'une base Oracle, et donc le plus petit élément de sauvegarde.

17.1 Créer un tablespace permanent

L'ordre SQL `CREATE TABLESPACE` permet de créer un tablespace :

```
CREATE [ BIGFILE | SMALLFILE ] TABLESPACE Ts nom
DATAFILE 'nom fichier' [ SIZE valeur [K|M|G|T] ] [ REUSE]
AUTOEXTEND { OFF | ON [ NEXT valeur [K|M|G|T] ]
[ MAXSIZE { UNLIMITED | valeur [K|M|G|T] } ] }
EXTENT MANAGEMENT
LOCAL { AUTOALLOCATE | UNIFORM [ SIZE valeur [K|M|G|T] ] }
SEGMENT SPACE MANAGEMENT
{ MANUAL | AUTO }
[ DEFAULT [ COMPRESS [ FOR { ALL | DIRECT_LOAD } OPERATIONS] | NOCOMPRESS ] storage ]
[ BLOCKSIZE valeur (K) ]
[ LOGGING | NOLOGGING ]
[ FORCE LOGGING ]
[ FLASHBACK { ON | OFF } ]
[ ONLINE | OFFLINE ]
;
```

- **DATAFILE** = spécification (emplacement, taille, ...) d'un (ou plusieurs) fichier(s) de données pour le tablespace, la taille peut être spécifiée en octets (pas de symbole), en Ko (symbole K) ou en Mo (symbole M) ou en Gigas (G) ou en terra pour les tablespaces de type **BIGFILE** uniquement.
- **AUTOEXTEND** = indique si le fichier de données peut (ON) ou non (OFF) grandir lorsque tout l'espace initialement alloué est occupé.
- **NEXT** = espace minimum alloué lors d'une extension
- **MAXSIZE** = taille maximum du fichier, éventuellement non limitée (UNLIMITED)
- **EXTENT MANAGEMENT** = permet de définir le mode de gestion des extents à l'intérieur du tablespace.
- **SEGMENT SPACE MANAGEMENT** = permet de définir le mode de gestion de l'espace libre des segments à l'intérieur du tablespace.
- **DEFAULT [COMPRESSE | NOCOMPRESS]** = permet de définir un mode de compression automatique des segments dans le tablespace, permettant de réduire l'espace disque. Attention, cette compression se fait au détriment des temps des performances.
- **LOGGING | NOLOGGING** = enregistre les segments dans les Redo Log si **LOGGING** (par défaut) est défini.
- **FORCE LOGGING** = oblige l'enregistrement des segments dans les Redo Log même si la clause **NOLOGGING** est définie.
- **FLASHBACK { ON | OFF }** = indique si le tablespace participe aux opérations de flashback (ON).
- **ONLINE | OFFLINE** = indique si le tablespace est accessible (ON) ou non (OFF).
- **COMPRESS | NOCOMPRESS** = compression des données contenues dans le tablespace et possibilité de créer une clause de stockage par défaut pour les segments créés sans clause de stockage.



D'une manière générale toutes les syntaxes peuvent être spécifiées en octets (pas de symbole), en kilo octets (K), en méga octets (M), en giga octet (G) ou encore Terra octet (T).

17.2 Modifier un tablespace permanent

Un certain nombre de manipulations sont effectuées durant la vie de ces tablespaces :

- ⇒ Allouer de l'espace supplémentaire à un tablespace
- ⇒ Passer un tablespace OFFLINE/ONLINE
- ⇒ Déplacer le(s) fichier(s) de données
- ⇒ Passer un tablespace READ ONLY/READ WRITE
- ⇒ Renommer un tablespace

Ces opérations s'effectuent avec la commande SQL :

⇒ ALTER TABLESPACE ou ALTER DATABASE.

17.2.1 Agrandir un tablespace

Ajouter un fichier de données

Il est possible d'agrandir un tablespace par l'ajout d'un fichier de données.

```
ALTER TABLESPACE nom
ADD DATAFILE spécification_fichier_data [...];
```

```
ALTER TABLESPACE data
ADD DATAFILE 'c:\app\guest\oradata\BORA\data02.dbf' SIZE 100M
AUTOEXTEND ON NEXT 100M MAXSIZE 500M;
```

Modifier la taille du fichier rattaché au tablespace

Modifier la taille d'un fichier de données, à la hausse ou à la baisse dans la limite du dernier extent occupé dans le fichier de données.

```
ALTER DATABASE
DATAFILE 'nom_complet' [...], RESIZE valeur [K|M];
```

L'option RESIZE peut être utilisée à la hausse mais aussi à la baisse en fonction de l'emplacement du dernier objet créé ou modifié, dans le tablespace, visible dans la vue DBA_EXTENTS, sinon il y a affichage d'un message d'erreur.

Cette option peut être lancée pendant que des utilisateurs travaillent sur la base. Elle a pour effet de modifier la taille du fichier immédiatement.

```
ALTER DATABASE
DATAFILE 'C:\app\oracle\oradata\BORA\data02.dbf' RESIZE 800M;
```



Modifier la clause AUTO EXTEND

L'option `AUTOEXTEND ON` a pour effet de permettre au fichier de s'étendre lorsqu'il est plein et qu'un nouvel objet est créé.

La clause `AUTOEXTEND OFF` annule une clause `AUTOEXTEND ON`.

```
ALTER DATABASE  
DATAFILE 'nom_complet' [...]  
AUTOEXTEND [ON|OFF]  
;
```

```
• désactivation de la clause autoextend  
ALTER DATABASE  
DATAFILE 'C:\app\oracle\oradata\BORA\data01.dbf'  
AUTOEXTEND OFF;  
  
• activation (ou modification) de la clause autoextend  
ALTER DATABASE  
DATAFILE 'C:\app\oracle\oradata\BORA\data02.dbf'  
AUTOEXTEND ON NEXT 200M MAXSIZE 800M;
```

17.2.2 Passer un tablespace OFFLINE ou ONLINE :

Les clauses `ONLINE` ou `OFFLINE` permettent d'activer ou de désactiver un tablespace. Les objets contenus dans le tablespace ne sont plus accessibles par les utilisateurs si la clause `OFFLINE` est utilisée.

```
ALTER TABLESPACE 'nom_complet'  
ONLINE | OFFLINE  
;
```

```
• désactivation  
ALTER TABLESPACE data OFFLINE;  
• activation  
ALTER TABLESPACE data ONLINE;
```

17.2.3 Passer un tablespace READ ONLY ou READ WRITE :

Le tablespace en `READ ONLY` ne permet plus l'écriture dans les objets qu'il contient.

```
ALTER TABLESPACE 'nom_complet'  
READ ONLY | READ WRITE  
;
```

```
• désactivation  
ALTER TABLESPACE data READ ONLY;  
• activation  
ALTER TABLESPACE data READ WRITE;
```



17.2.4 Déplacer un fichier de données

Cette action est possible par un `ALTER TABLESPACE` ou un `ALTER DATABASE RENAME FILE`.

Le `ALTER DATABASE` est particulièrement utile pour déplacer le tablespace `SYSTEM` qui ne peut pas être mis `OFFLINE`.



Le tablespace concerné doit être OFFLINE ou la base en état MOUNT dans le cas du ALTER DATABASE.

Les commandes ne manipulent pas physiquement le fichier ;

Elles se contentent de mettre à jour le dictionnaire Oracle et les fichiers de contrôle, de plus le fichier doit être renommé + copié + déplacé avant de passer la commande.

```
ALTER TABLESPACE nom
RENAME DATAFILE 'ancien nom complet'
TO 'nouveau nom complet'
;
```

Le mode opératoire ci-dessous doit être appliqué pour effectuer ce genre d'opération.

MODE OPERATOIRE



- Mettre le tablespace OFFLINE
`ALTER TABLESPACE nom_ts OFFLINE ;`
- Déplacer les fichiers vers le nouvel emplacement
`COPIER .. +.. COLLER`
- Indiquer à Oracle le nouvel emplacement
`ALTER TABLESPACE RENAME DATAFILE ancien_nom
TO nouveau_nom ;`
- Mettre le tablespace ONLINE
`ALTER TABLESPACE nom_ts ONLINE ;`


Déplacer un fichier de données

- Passer le tablespace OFFLINE
`Sql> alter tablespace data OFFLINE ;`
- Déplacer le fichier du tablespace par une commande du système d'exploitation (copy)
- Exécuter la commande : `alter tablespace ...`
`Sql> alter tablespace data`
`Sql> rename datafile 'C:\app\guest\oradata\BORA\data01.dbf'`
`Sql> to 'e:\oracle\oradata\BORA\data01.dbf' ;`
- Remettre le tablespace ONLINE




```
|Sql> alter tablespace data ONLINE;
```

Le mode opératoire ci-dessous doit être appliqué pour effectuer ce genre d'opération sur le tablespace SYSTEM qui contient le dictionnaire de données et ne peut pas être mis OFFLINE.

MODE OPERATOIRE	
	⇒ Arrêter la base SHUTDOWN IMMEDIATE
	⇒ Déplacer les fichiers vers le nouvel emplacement COPIER ... +... COLLER
	⇒ Démarrer la base à l'état MOUNT STARTUP MOUNT
	⇒ Indiquer à Oracle le nouvel emplacement ALTER DATABASE RENAME FILE ancien_nom TO nouveau_nom
	⇒ Ouvrir la base ALTER DATABASE OPEN

Déplacer le tablespace système

```
Le mettre à l'état MOUNT  
Sql> Connect / as sysdba  
Sql> Shutdown immediate  
Sql> Startup mount
```

Déplacer le fichier du tablespace système par une commande du système d'exploitation (copy)

```
Exécuter la commande : alter database rename file ...  
Sql> alter database  
Sql> rename file 'C:\app\oracle\oradata\BORA\systeme01.dbf'  
Sql> to 'D:\app\oracle\oradata\BORA\systeme01.dbf' ;
```

```
Ouvrir la base  
Sql> alter database open;
```

17.2.5 Renommer un tablespace

A partir de la version 10g, il est possible de changer le nom d'un tablespace en utilisant une requête SQL ou OEM. Quand vous renommez un tablespace, Oracle met à jour toutes les références au nom du tablespace dans le dictionnaire de données, le fichier de contrôle et les en-têtes de fichiers du tablespace concerné.



L'identifiant du tablespace n'est pas changé. Si le tablespace s'avère être le tablespace par défaut d'un utilisateur, alors le tablespace renommé apparaît toujours comme étant le tablespace par défaut de l'utilisateur.

Les en-têtes de fichiers ne peuvent être changés que si le tablespace est mis dans un mode `READ WRITE` (quand c'est possible).

Vous pouvez renommer à la fois des tablespaces permanents et temporaires.

```
ALTER TABLESPACE 'ancien nom' RENAME TO 'nouveau nom' ;
```

```
SQL> -- Renommer le tablespace DATABIS en DATA
SQL> ALTER TABLESPACE databis RENAME TO data
2 /
Tablespace altered.
```

Si vous recréez le fichier de contrôle, les « *records* » de tablespace dans le nouveau fichier de contrôle sont construits à partir des informations des fichiers associés au tablespace.

Si vous utilisez des sauvegardes pour la restauration, les « *records* » de tablespace construits pourraient refléter les anciens noms du tablespace car la sauvegarde a été effectuée avant que le tablespace soit renommé.

La restauration en utilisant des sauvegardes de fichiers de données contenant des anciens noms de tablespace n'est pas un problème. Si un fichier de sauvegardes dont les en-têtes contiennent l'ancien nom du tablespace est restauré après avoir été renommé, l'en-tête de fichier possède le nouveau nom du tablespace après restauration.



Il n'est pas possible de renommer les tablespaces SYSTEM, SYSAUX ou UNDO. Le changement affecterait à la fois la mémoire et le SPFILE

Si un seul datafile du tablespace à renommer est `OFFLINE` ou si le tablespace est `OFFLINE`, alors le tablespace ne peut pas être renommé.

Avant la version 10g, si vous vouliez réorganiser et migrer un Tablespace « DATA » géré par le dictionnaire en un tablespace géré localement avec des extents uniformes,

vous deviez appliquer le mode opératoire ci-dessous :

- ◆ Créer un autre ts « DATABIS »,
- ◆ Copier tous les objets de « DATA » vers « DATABIS »
- ◆ Supprimer « DATA »
- ◆ Recréer « DATA » avec une gestion locale des extents,
- ◆ Copier tous les objets de « DATABIS » vers « DATA »,
- ◆ Supprimer « DATABIS ».



En utilisant la fonctionnalité de renommage d'un tablespace, vous pouvez simplifier ce mode opératoire :

- ◆ Créer un ts « DATABIS » géré localement
- ◆ Copier tous les objets de « DATA » vers « DATABIS »
- ◆ Supprimer « DATA »
- ◆ Renommer « DATABIS » en « DATA »

Le tablespace ne peut être renommé que si vous avez déjà supprimé le tablespace portant l'ancien nom.



Comme Oracle ne renomme pas les fichiers d'un tablespace, il faut faire attention aux noms des fichiers associés au tablespace. Faites attention à la méthode qui consistait à nommer les fichiers d'un tablespace du même nom suivi de numéros car après « renommage », ces noms deviennent obsolètes.

Cette fonctionnalité est à utiliser en cas de migration de oracle 8i vers une version oracle 10g.

17.2.6 Supprimer un tablespace

L'ordre SQL DROP TABLESPACE permet de supprimer un tablespace.

```
DROP TABLESPACE nom [ INCLUDING CONTENTS [ AND DATAFILES ]  
[ CASCADE CONSTRAINTS ] ]  
;
```

Si le tablespace contient des segments, la clause INCLUDING CONTENTS est nécessaire.

La clause CASCADE CONSTRAINTS permet en plus d'ignorer les contraintes d'intégrité référentielle définies sur des tables hors du tablespace et qui référencent les clés primaires des tables à l'intérieur du tablespace.

A partir de la 9i, la clause AND DATAFILES permet de supprimer les fichiers physiquement.

Un message est écrit dans le fichier des alertes de l'instance pour chaque fichier supprimé.

```
Alter tablespace data OFFLINE ;  
DROP TABLESPACE data INCLUDING CONTENTS and DATAFILES;
```



Ne pas oublier de mettre le tablespace OFFLINE avant de le supprimer.



17.2.7 Créer un tablespace avec une taille de bloc non standard

La taille de bloc standard d'un tablespace correspond à la taille définie par le paramètre `DB_BLOCK_SIZE` lors de la création de la base de données. Cette taille ne peut plus être modifiée par la suite (pendant toute la vie de la base de données).

⇒ elle est utilisée pour le tablespace `SYSTEM` et les tablespaces créés à la création de la base tels que `SYSAUX`, `UNDO`, `TEMP` ou `USERS`

A partir d'Oracle 9i, il est possible d'utiliser plusieurs tailles de bloc dans la base de données.

Jusqu'à 5 autres tailles de bloc peuvent être utilisées.

Les valeurs permises sont 2 ko, 4 ko, 8 ko, 16 ko et 32 ko (certaines plates-formes sont plus restrictives), elles sont utilisées lors de la création des autres tablespaces.

Pour les utiliser, il suffit de configurer des sous-caches correspondants dans le *Database Buffer Cache*.



Un tablespace ne peut pas être créé avec une taille de bloc non standard si le cache correspondant n'est pas configuré auparavant (DATABASE BUFFER CACHE).

La colonne `BLOCK_SIZE` de la vue `DBA_TABLESPACES` donne la taille de bloc utilisée par les tablespaces. Cette possibilité d'utiliser plusieurs tailles de bloc est surtout intéressante pour la fonctionnalité de transport de tablespace.

Ainsi un tablespace ayant une taille de bloc de 4 ko peut être transportée dans une base utilisant des blocs de 8 ko.

La taille de bloc d'un tablespace ne peut pas être modifiée sans recréer le tablespace.

17.2.8 Le cryptage des données d'un tablespace (nouveau 11g)

La version 11g permet de crypter la totalité des données contenues dans un tablespace. Cette option de la base de données est disponible en version Enterprise Edition et est payante.

Elle permet d'utiliser un algorithme de cryptage disponible dans la base de données.

Il existe différents algorithmes de cryptage :

- ◆ 3DES168
- ◆ AES128 (par défaut)
- ◆ AES192
- ◆ AES256



Pour utiliser un algorithme de cryptage il faut le préciser au moment de la création du tablespace :

```
Create tablespace my_secure_tbs  
Datafile '/oracle/oradata/Tahiti/my_secure_tbs01.dbf' size 100M  
Encryption using '3DES168' default storage (encrypt) ;
```

Si vous utilisez l'algorithme par défaut il n'est pas nécessaire de préciser l'algorithme :

```
Create tablespace my_secure_tbs  
Datafile '/oracle/oradata/Tahiti/my_secure_tbs01.dbf' size 100M  
Encryption default storage (encrypt) ;
```



En version 11g, si on utilise le cryptage des données d'un tablespace, il faut faire attention à ce que les données cryptées ne soient pas plus longues que la taille des colonnes définies pour les tables !
par exemple une colonne définie en VARCHAR(4000) peut avoir un contenu non crypté à 3899 et il devient supérieur à 4000 lorsqu'il est crypté.

Le cryptage des données dans un tablespace a des conséquences significatives sur les performances de la base de données. Il faut donc manipuler cette option de la base de données avec précaution.

17.2.9 Tablespace de travail par défaut

Chaque utilisateur de la base de données possède un tablespace permanent pour stocker des données permanentes et un tablespace temporaire pour les données temporaires (tris),

Oracle vous permet de définir un tablespace permanent par défaut qui est automatiquement utilisé à chaque fois qu'un nouvel utilisateur est créé sans tablespace spécifique permanent.

Le nouveau concept d'un tablespace permanent par défaut ne s'applique pas aux USERS system (SYS, SYSTEM, OUTLN). Ces USERS utilisent toujours le tablespace SYSTEM comme leur tablespace permanent.



Vous ne pouvez pas supprimer un tablespace désigné comme tablespace permanent par défaut. Vous devez d'abord réassigner un autre tablespace aux utilisateurs en tant que tablespace par défaut puis supprimer l'ancien tablespace par défaut.

Il est possible de modifier le tablespace permanent par défaut de la base de données. Le changement prend effet pour les nouveaux objets ou utilisateurs créés après la commande ALTER DATABASE.

Après l'exécution de cette commande tous les utilisateurs non « SYSTEM » sont rattachés au tablespace « newusers ».

```
ALTER DATABASE DEFAULT TABLESPACE 'newusers' ;
```



```
SQL> alter database default tablespace users
2 /
Database altered.
SQL> -- Afficher les tablespaces par défaut
SQL> select property_value, property_name
2 from database_properties
3 where property_name like 'DEFAULT%'
4 /
```

PROPERTY_VALUE	PROPERTY_NAME
TEMP	DEFAULT_TEMP_TABLESPACE
USERS	DEFAULT_PERMANENT_TABLESPACE
SMALLFILE	DEFAULT_TBS_TYPE

17.2.10 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les tablespaces et les fichiers de données :

- DBA_TABLESPACES : informations sur les tablespaces
- DBA_DATA_FILES : informations sur les fichiers de données
- V\$TABLESPACE : informations sur les tablespaces (à partir du fichier de contrôle)
- V\$DATAFILE : informations sur les fichiers de données (à partir du fichier de contrôle)
- V\$SYSAUX_OCCUPANTS : informations sur les objets des occupants stockés dans le tablespace SYSAUX.
- DBA_FREE_SPACE : informations sur l'espace disponible à l'intérieur d'un tablespace, un tablespace qui n'a pas d'extent libre n'a pas de ligne dans DBA_FREE_SPACE.
- DBA_SEGMENTS : informations sur les segments alloués à l'intérieur d'un tablespace
- DBA_EXTENTS : informations sur les extents alloués à l'intérieur d'un tablespace



18 Tablespaces SYSTEM & SYSAUX

18.1 Tablespace SYSTEM

Le tablespace `SYSTEM` est le tablespace qui contient le dictionnaire de données.

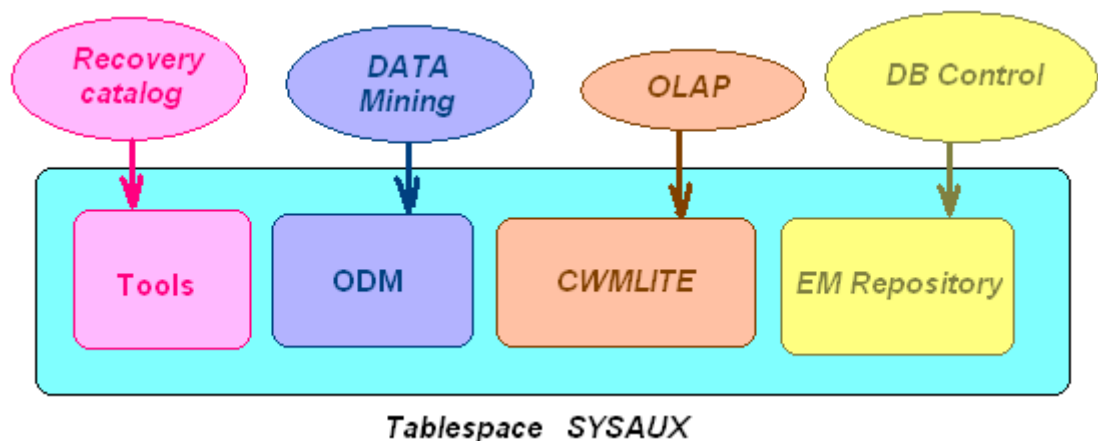
Le dictionnaire de données est installé à la création de la base de données dans le tablespace `SYSTEM` par l'utilisateur `sys`.

Pour des raisons de performance et de taille de stockage, le tablespace `SYSTEM` ne contient plus d'autres occupants à partir de la version Oracle 10g. Ces occupants sont placés dans le tablespace `SYSAUX`.

18.2 Tablespace SYSAUX

Le tablespace `SYSAUX` est un tablespace auxiliaire au tablespace `SYSTEM`.

Des composants utilisaient dans les versions antérieures d'Oracle le tablespace `SYSTEM` ou leur propre tablespace. Depuis la version 10g, ces composants utilisent le tablespace `SYSAUX` comme destination par défaut pour stocker leurs données.



Ce tablespace est obligatoire et doit être créé au moment de la création de la base de données ou de son *upgrade*.

Ce tablespace `SYSAUX` est défini avec les attributs :

- ♦ Permanent
- ♦ Read Write
- ♦ Extent Management Local
- ♦ Segment Space Management Auto

Comme pour le tablespace `SYSTEM`, il est possible de créer vos propres tables à l'intérieur du tablespace `SYSAUX` ce qui est vivement déconseillé.

Ce nouveau tablespace fournit un emplacement centralisé pour toutes les « métas data » auxiliaires de la base de données qui ne résident pas dans le tablespace `SYSTEM`. Il réduit ainsi le nombre de tablespaces créés par défaut.



Le tablespace `SYSAUX` n'est pas un tablespace transportable.



Vous ne devez pas supprimer ou renommer le tablespace `SYSAUX`.
Si le tablespace `SYSAUX` n'est pas disponible (`ENABLE`) certaines fonctionnalités de la base de données ne fonctionneront plus.

18.2.1 Avantages du tablespace `SYSAUX`

⇒ Réduction du nombre de tablespaces :

Plusieurs fonctionnalités d'Oracle nécessitent un tablespace spécifique pour stocker les données. Le plus souvent un tablespace séparé est créé. En conséquence, les DBA se retrouvent dans l'obligation de gérer un grand nombre de tablespaces. Le tablespace `SYSAUX` est le tablespace par défaut pour ces produits Oracle en utilisant le tablespace `SYSAUX`, le travail du dba est simplifié.

⇒ Réduction de la charge sur le tablespace `SYSTEM` :

Certains produits ou fonctionnalités Oracle utilisent le tablespace `SYSTEM` comme tablespace par défaut pour stocker les données. De ce fait, les performances de la base sont dégradées à cause d'un large volume de données dans le tablespace `SYSTEM`. De plus, cela peut affecter la disponibilité de la base à cause des risques de corruption et de manque d'espace disque au fur et à mesure que celui-ci grandit.

⇒ Gestion simplifiée du RAC :

Pour les utilisateurs du RAC avec des raw devices (périphérique), un raw device doit être alloué pour chaque tablespace créé. Gérer un grand nombre de raw device peut être difficile. Regrouper ces tablespaces dans le tablespace `SYSAUX` réduit le nombre de raw device qu'un DBA doit allouer.

Le tablespace `SYSAUX` peut être modifié avec la commande « `ALTER TABLESPACE` ». Il n'y a pas de changement de syntaxe pour le tablespace `SYSAUX`.

Vous pouvez réaliser des tâches d'administration classiques telles que : ajouter des fichiers de données au tablespace `SYSAUX`.

Vous devez avoir le privilège `SYSDBA` pour modifier le tablespace `SYSAUX`.

18.2.2 Délocaliser les occupants du tablespace `SYSAUX`

Après la création vous pouvez surveiller l'utilisation de l'espace de chaque occupant à l'intérieur du tablespace `SYSAUX` en utilisant la console `OEM`. Si vous détectez qu'un composant prend trop d'espace dans le tablespace `SYSAUX`, vous pouvez déplacer l'occupant dans un tablespace différent.

- Déplacer les données d'OLAP
- du tablespace `sysaux` vers le nouveau tablespace `CWMLITE`.

```
Select occupant_name, space_usage_kbytes  
From v$sysaux_occupants ;
```




```
Select occupant_name, schema_name, move_procedure  
From v$sysaux occupants ;  
  
Exec WKSYS.MOVE_WK('CWMLITE');  
Exec WKSYS.MOVE_WK('SYSAUX');
```

La seconde requête est utilisée pour déterminer quelle procédure doit être utilisée pour déplacer l'occupant correspondant hors du tablespace `SYSAUX`. Cette procédure gère les deux directions, entrant et sortant du tablespace `SYSAUX`.

⇒ Les procédures fonctionnent base `ONLINE`.

Les occupants suivants ne peuvent pas être déplacés :

⇒ `STREAMS`, `SMC`, `STATSPACK`, `ORDIM`, `ORDIM/PLUGINS`, `ORDIM/SQLMM`, `JOB_SCHEDULER`.



19 Tablespace UNDO

Depuis la version 9i, le tablespace UNDO permet la gestion automatique des segments d'annulation. Cette fonctionnalité est appelée *Automatic Undo Management* (AUM) ou encore *System Managed Undo* (SMU).



Utiliser la gestion MANUELLE des rollbacks segments en version 11g peut provoquer de graves problèmes de performance !

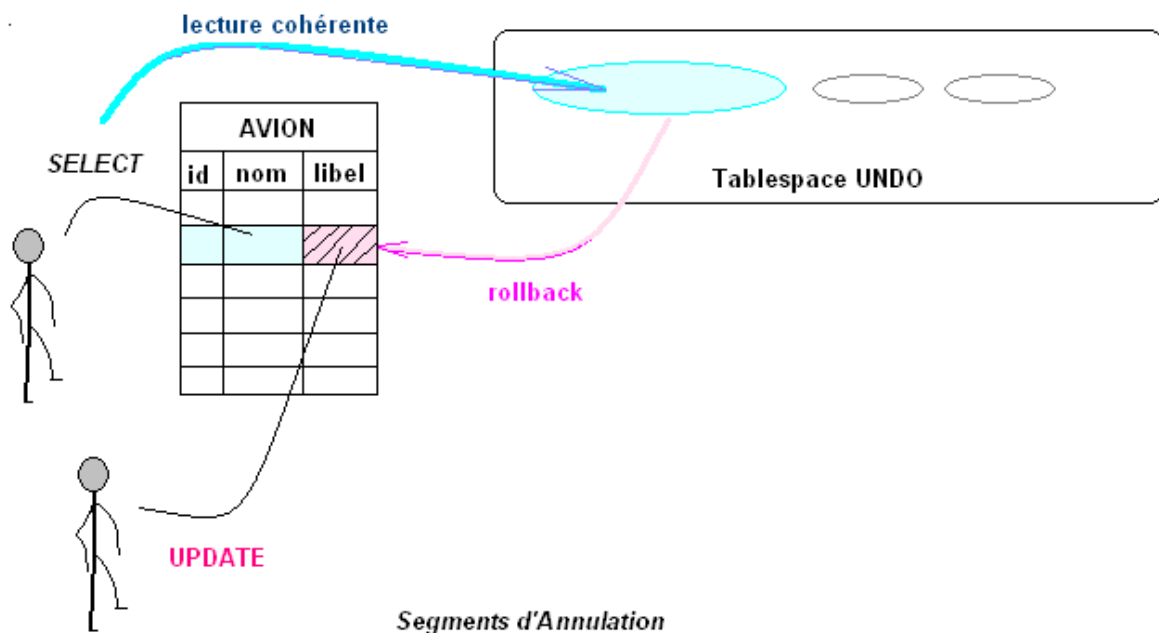
Un segment d'annulation est un segment utilisé pour stocker la version précédente (« *image avant* ») des données en cours de modification dans une transaction qui n'est pas encore validée (non COMMITée).

- ⇒ Permet l'annulation de la transaction (ROLLBACK)
- ⇒ Permet la lecture cohérente (même version des données lues dans un SELECT)
- ⇒ Certaines fonctionnalités de FLASHBACK
- ⇒ En cas de RECOVER pour annuler des modifications non commitées

Si la transaction est validée (COMMIT), l'espace sera libéré, par contre si elle est annulée (ROLLBACK) la version précédente des données sera réécrite.

La lecture cohérente est le fait que les données en cours de modification ne sont pas vues par les autres utilisateurs tant que la transaction n'est pas validée.

Lorsqu'un utilisateur interroge une table en cours de modification, Oracle utilise l'image avant des données stockées dans les segments d'annulation pour lui répondre.



Avant la version 9i les segments d'annulation n'existaient pas, les transactions étaient gérées par des *rollbacks segments*. Les *rollbacks segments* étaient gérés par les DBA.

Aujourd'hui encore une base de données a toujours au moins un *rollback segment* nommé `SYSTEM`, réservé aux transactions qui concernent les objets du tablespace `SYSTEM`, stocké dans le tablespace `SYSTEM`. Ce *rollback segment* est utilisé pour les accès au dictionnaire de données lors de la création de la base de données.

D'autres segments d'annulation sont nécessaires pour les transactions qui concernent les objets contenus dans les tablespaces (tables).



Un segment d'annulation est stocké dans un tablespace, composé d'extents dont les blocs sont chargés dans le Database Buffer Cache en fonction des besoins. A partir de la version 9i, le tablespace UNDO remplace les *rollbacks segments*. Il contient des segments d'annulation gérés par Oracle.

Les segments d'annulation sont constitués d'extents.

Les modifications apportées aux blocs des segments d'annulation dans le *Database Buffer Cache* sont enregistrés dans les fichiers du tablespace UNDO si l'instance a besoin d'espace en mémoire vive.

Ils sont également enregistrés dans les redo Logs. Ainsi, en cas de restauration d'instance ou de restauration de média (restauration d'un tablespace), Oracle est en mesure de reconstituer les segments d'annulation et d'annuler les modifications déjà écrites dans les fichiers de données pour annuler les transactions non validées au moment de l'incident.

Les extents d'un segment d'annulation sont utilisés de la façon suivante :

- ♦ Les transactions écrivent dans l'extent courant
- ♦ Lorsque l'extent courant est plein, les transactions passent au suivant si celui-ci est libre (pas de transaction active à l'intérieur)
- ♦ Si l'extent suivant n'est pas libre, un nouvel extent est alloué au segment d'annulation

Par défaut, c'est Oracle qui alloue les segments d'annulation aux transactions en cherchant à répartir les transactions concurrentes sur les différents segments d'annulation.

Lorsqu'une transaction commence à utiliser un segment d'annulation, elle ne peut pas changer de segment en cours de déroulement. Elle inscrit son identifiant dans l'en-tête du segment puis utilise les blocs dont elle a besoin dans le segment.

Lorsque la transaction se termine, elle libère le segment d'annulation mais les informations de rollback ne sont pas supprimées immédiatement.

Ces informations peuvent encore être utiles pour une lecture cohérente.

- ⇒ Ces informations sont conservées pendant toute la durée de rétention définie par le paramètre `UNDO_RETENTION`.





**Si le paramètre UNDO_RETENTION est trop faible (900 par défaut), l'erreur « snapshot too old » apparaît.
➔ il faut augmenter la durée de rétention !**

Un segment d'annulation peut contenir plusieurs transactions simultanées, les extents pouvant même être partagés entre plusieurs transactions (mais pas les blocs).

Une transaction bloquée peut bloquer un extent et obliger le segment à grossir alors que d'autres extents sont libres derrière l'extent bloqué ; cette situation peut être détectée grâce à la vue V\$TRANSACTION.



Lorsqu'un segment d'annulation grossit et atteint la limite de la taille du tablespace UNDO, il faut retailler le tablespace UNDO puis relancer la transaction.

19.1 Fonctionnement du tablespace UNDO

Dans un tablespace d'annulation les segments d'annulation sont automatiquement gérés par Oracle et lui seul.

- ◆ Nommés _SYSSMU n \$
- ◆ Dimensionnés automatiquement en fonction des besoins (nombre et taille).
- ◆ S'étendent et rétrécissent en fonction des besoins.

```
SQL> COL name FOR A10
SQL> SELECT n.name,s.extends,s.shrinks,s.wraps,s.hwmsize,s.writes
2  FROM v$rollstat s,v$rollname n
3  WHERE n.usn = s.usn
4  ORDER BY n.usn
5  /
```

NAME	EXTENDS	SHRINKS	WRAPS	HWMSIZE	WRITES
SYSTEM	0	0	0	385024	4520
_SYSSMU1\$	0	0	0	1040384	7176
_SYSSMU2\$	0	0	0	1171456	108
_SYSSMU3\$	0	0	0	1236992	160
_SYSSMU4\$	0	0	0	1499136	160
_SYSSMU5\$	0	0	0	1302528	160
_SYSSMU6\$	0	0	0	253952	236
_SYSSMU7\$	0	0	0	319488	54
_SYSSMU8\$	0	0	0	450560	54
_SYSSMU9\$	0	0	0	1564672	54
_SYSSMU10\$	0	0	0	319488	160

```
11 ligne(s) sélectionné(s).
```

Les segments d'annulation stockés dans le tablespace d'annulation sont automatiquement activés.



Il est impossible de toucher aux segments d'annulation (ajout, suppression, activation, dimensionnement ...), c'est Oracle qui les gère.

Lors de la création du tablespace d'annulation, Oracle crée 10 segments d'annulation de 2 extents chacun.

En fonction des besoins, Oracle alloue des extents supplémentaires aux segments d'annulation existants ou crée des segments d'annulation supplémentaires (un par un).

Les segments d'annulation créés ne sont pas supprimés. S'il y a baisse d'activité transactionnelle, Oracle n'en a plus l'utilité, il les passe `OFFLINE`. Si l'instance en a besoin ultérieurement, elle les repassera `ONLINE`.

19.2 Positionner les paramètres de gestion automatique

La gestion automatique des segments d'annulation se fait lors de la création de la base de données.

La gestion automatique des segments UNDO fait appel aux paramètres suivants :

- ♦ `UNDO_MANAGEMENT` : mode gestion souhaité pour les informations d'annulation (par défaut à `TRUE`).
- ♦ `UNDO_RETENTION` : durée de rétention/conservation des informations d'annulation dans les segments d'annulation (en seconde), 900 par défaut (soit ¼ d'heure).
- ♦ `UNDO_TABLESPACE` : nom du tablespace d'annulation à utiliser, si non spécifié Oracle prend le premier tablespace UNDO disponible.

```
SQL> show parameters undo
NAME                                TYPE                                VALUE
-----                                -                                -
undo_management                     string                             AUTO
undo_retention                      integer                           900
undo_tablespace                     string                             UNDOTBS
```

Si le paramètre d'initialisation `UNDO_MANAGEMENT` est positionné à `AUTO` et que la clause `UNDO TABLESPACE` n'est pas présente, Oracle crée un tablespace d'annulation par défaut :

- ⇒ Nommé `SYS_UNDOTBS`
- ⇒ Avec un fichier de données de 10 Mo, positionné en `AUTOEXTEND`

19.3 Créer un tablespace UNDO

La création d'un tablespace UNDO se fait lors de la création de la base de données et s'effectue grâce à la clause `UNDO_TABLESPACE` de l'ordre `SQL CREATE DATABASE`.

Le tablespace d'annulation est forcément géré localement, avec une gestion automatique des extents, la clause `EXTENT MANAGEMENT` peut être indiquée, mais la seule valeur autorisée est `LOCAL AUTOALLOCATE`.



Si un nom a été indiqué dans le paramètre d'initialisation `UNDO_TABLESPACE`, utilisez le même nom dans la clause `UNDO TABLESPACE`, sinon :

- ⇒ La base sera bien créée avec le tablespace spécifié dans la clause `UNDO TABLESPACE`
- ⇒ Mais Oracle retournera une erreur à l'ouverture de celle-ci

19.4 Changer de tablespace UNDO actif

19.4.1 Créer un tablespace UNDO après création de la base

S'effectue grâce à l'ordre SQL `CREATE UNDO TABLESPACE` :

```
CREATE UNDO TABLESPACE nom
DATAFILE spécification_fichier_data [,...]
;
```

```
SQL> CREATE UNDO TABLESPACE undotbs_essai
2 DATAFILE 'C:\app\guest\oradata\BORA\undotbs_essai01.dbf'
3 SIZE 10M AUTOEXTEND ON NEXT 20M MAXSIZE 50M
4 /
```

Tablespace créé.

```
SQL> select TABLESPACE_NAME,STATUS,SEGMENT_NAME
2 from dba_rollback_segs;
```

TABLESPACE_NAME	STATUS	SEGMENT_NAME
SYSTEM	ONLINE	SYSTEM
UNDOTBS	ONLINE	_SYSSMU1\$
UNDOTBS	ONLINE	_SYSSMU2\$
UNDOTBS	ONLINE	_SYSSMU3\$
UNDOTBS	ONLINE	_SYSSMU4\$
UNDOTBS	ONLINE	_SYSSMU5\$
UNDOTBS	ONLINE	_SYSSMU6\$
UNDOTBS	ONLINE	_SYSSMU7\$
UNDOTBS	ONLINE	_SYSSMU8\$
UNDOTBS	ONLINE	_SYSSMU9\$
UNDOTBS	ONLINE	_SYSSMU10\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU11\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU12\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU13\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU14\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU15\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU16\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU17\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU18\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU19\$
UNDOTBS_ESSAI	OFFLINE	_SYSSMU20\$

```
21 ligne(s) sélectionné(s).
```



19.4.2 Changer de tablespace UNDO pendant l'activité de la base

Si la base dispose de plusieurs tablespaces d'annulation, il est possible de changer de tablespace actif en modifiant la valeur du paramètre `UNDO_TABLESPACE` qui est un paramètre dynamique :

```
ALTER SYSTEM SET UNDO TABLESPACE = nouveau nom [ clause SCOPE ]  
;
```

Lors d'un changement de tablespace UNDO actif, les segments d'annulation stockés dans l'ancien tablespace d'annulation sont désactivés (passés `OFFLINE`) et les segments d'annulation stockés dans le nouveau tablespace d'annulation sont activés (passés `ONLINE`).

Le changement de tablespace d'annulation actif n'attend pas que les transactions en cours se terminent.

Oracle met les segments d'annulation utilisés dans le statut `PENDING OFFLINE` (visible uniquement dans `V$ROLLSTAT`), empêchant ainsi qu'ils soient utilisés par de nouvelles transactions ; les segments d'annulation sans transaction active sont immédiatement passés `OFFLINE`.

Les segments d'annulation `PENDING OFFLINE` sont passés définitivement `OFFLINE` lorsqu'ils ne contiennent plus de transaction active.

Les nouvelles transactions utilisent les segments d'annulation du nouveau tablespace d'annulation actif.

```
SQL> alter system set undo_tablespace=undotbs_essai scope=memory;  
Système modifié.
```

```
SQL> select TABLESPACE_NAME,STATUS,SEGMENT_NAME  
2 from dba_rollback_segs;
```

TABLESPACE_NAME	STATUS	SEGMENT_NAME
SYSTEM	ONLINE	SYSTEM
UNDOTBS	OFFLINE	_SYSSMU1\$
UNDOTBS	OFFLINE	_SYSSMU2\$
UNDOTBS	OFFLINE	_SYSSMU3\$
UNDOTBS	OFFLINE	_SYSSMU4\$
UNDOTBS	OFFLINE	_SYSSMU5\$
UNDOTBS	OFFLINE	_SYSSMU6\$
UNDOTBS	OFFLINE	_SYSSMU7\$
UNDOTBS	OFFLINE	_SYSSMU8\$
UNDOTBS	OFFLINE	_SYSSMU9\$
UNDOTBS	OFFLINE	_SYSSMU10\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU11\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU12\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU13\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU14\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU15\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU16\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU17\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU18\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU19\$
UNDOTBS_ESSAI	ONLINE	_SYSSMU20\$

```
21 ligne(s) sélectionné(s).
```



Si des transactions sont en cours dans l'ancien tablespace au moment du changement alors :

- ♦ Le changement est pris en compte immédiatement : les nouvelles transactions utilisent le nouveau tablespace d'annulation mais Oracle laisse les transactions actuelles se terminer dans l'ancien
- ♦ Les segments d'annulation non utilisés de l'ancien tablespace sont passés `OFFLINE`
- ♦ Les segments d'annulation utilisés de l'ancien tablespace sont marqués `PENDING OFFLINE` (dans `V$ROLLSTAT`)
- ♦ Lorsque les transactions sont terminées, les segments d'annulation utilisés sont définitivement passés `OFFLINE`
- ♦ L'ancien tablespace reste `ONLINE` : il n'est simplement plus le tablespace actif pour l'annulation (défini par le paramètre `UNDO_TABLESPACE`)



Un tablespace qui contient des segments d'annulation `ONLINE` ou `PENDING OFFLINE` ne peut pas être désactivé ou supprimé.

Bien noter que l'ancien tablespace d'annulation actif reste `ONLINE`, mais que les segments d'annulation qu'il contient sont `OFFLINE` (ils ne peuvent pas être passés `ONLINE` à la main) ; si besoin, il faut passer le tablespace d'annulation explicitement `OFFLINE`.

Activer ou désactiver le tablespace, uniquement s'il n'est pas le tablespace d'annulation **actif** et s'il ne contient pas de segments d'annulation `PENDING OFFLINE`



Le passage `OFFLINE` d'un tablespace d'annulation actif, ou contenant encore des segments d'annulation `PENDING OFFLINE`, est interdit.

19.5 Administrer un tablespace UNDO

S'effectue avec les ordres SQL habituels `ALTER TABLESPACE` ou `ALTER DATABASE` (pour la gestion des fichiers de données) comme un tablespace permanent ordinaire, mais restreints aux actions suivantes :

- ♦ Activer ou désactiver le tablespace (comme présenté ci-dessus)
- ♦ Ajouter un fichier de données
- ♦ Modifier la taille ou l'extension automatique d'un fichier de données
- ♦ Déplacer un fichier de données

Voir page 94.



19.5.1 Dimensionner le tablespace UNDO

Utiliser la vue V\$UNDOSTAT pour dimensionner le tablespace d'annulation.

Les besoins en espace d'annulation sont liés à la durée de rétention des informations (paramètre UNDO_RETENTION).

La quantité totale d'espace d'annulation nécessaire pour satisfaire la durée de rétention peut être estimée par la formule.

⇒ $\text{UNDO_RETENTION} * \text{Quantité d'espace d'annulation par seconde}$

Le paramètre UNDO_RETENTION peut être défini en analysant la valeur de la colonne MAXQUERYLEN de la vue V\$UNDOSTAT.

⇒ Donne la durée en seconde de la requête la plus longue sur chaque période analysée

La quantité d'espace d'annulation par seconde peut être estimée en analysant la valeur de la colonne UNDOBLKS de la vue V\$UNDOSTAT.

⇒ Donne le nombre de blocs d'annulation utilisés sur chaque période analysée

Pour que le calcul soit pertinent, il est important de faire une analyse en pleine charge.

Exemple

```
3000 blocs utilisés en pleine charge, sur 10' ( Soit 3000 / 600 = 5 blocs par seconde
Durée de rétention = 30' = 1800 seconde ( Soit un besoin de 5 * 1800 = 9000 blocs
Taille de bloc = 4 K0 ( Soit une taille de 4 * 9000 / 1024 = environ 35 Mo
Prendre de la marge ( 50 Mo)
```

Exemple 2

```
SQL> SET LINESIZE 75
SQL> SET TRIMSPOOL ON
SQL> SET PAGESIZE 1000
SQL> SET VERIFY OFF
SQL> COL begin_time FOR A11
SQL> COL end_time FOR A11

SQL> -- interrogation de V$UNDOSTAT
SQL> SELECT
2      TO_CHAR(begin_time,'DD/MM HH24:MI') begin_time,
3      TO_CHAR(end_time,'DD/MM HH24:MI') end_time,
4      undoblks,
5      maxquerylen
6 FROM    v$undostat
7
8 /
BEGIN_TIME      END_TIME      UNDOBLKS  MAXQUERYLEN
-----
14/08 11:03      14/08 11:09           0           3
14/08 10:53      14/08 11:03        526          77
14/08 10:43      14/08 10:53        601         192
14/08 10:33      14/08 10:43        635          72
14/08 10:23      14/08 10:33        666         147
14/08 10:13      14/08 10:23        609          84
```



```
14/08 10:03    14/08 10:13          618          65
14/08 09:53    14/08 10:03          150          44
14/08 09:43    14/08 09:53        1000          17
14/08 09:33    14/08 09:43          374           1
14/08 09:23    14/08 09:33           2           1

11 rows selected.

SQL> -- durée de la requête la plus longue, toutes périodes confondues
SQL> -- * bonne base de départ pour le paramètre UNDO_RETENTION

SQL> SELECT  MAX(maxquerylen) maxquerylen
3 FROM    v$undostat
5 /

MAXQUERYLEN
-----
192
SQL> -- plus grand nombre de blocs d'annulation utilisés,
SQL> -- toutes périodes confondues
SQL> SELECT  MAX(undoblks) undoblks,
3          MAX(undoblks) / 600 « UNDOBLKS/SECONDE »
4 FROM    v$undostat
6 /

UNDOBLKS UNDOBLKS/SECONDE
-----
1000          1,666666667
SQL> -- estimation des besoins minimums en espace d'annulation
SQL> SELECT MAX(maxquerylen)*(MAX(undoblks) / 600)*block_size_ko « Taille (ko) »
4 FROM    v$undostat,
6          (SELECT value/1024 block_size_ko FROM v$parameter
7 WHERE name = 'db_block_size') p
8 GROUP BY block_size_ko
10 /

Taille (ko)
1280
```

19.5.2 Supprimer un tablespace UNDO

S'effectue avec l'ordre SQL habituel `DROP TABLESPACE`, la clause `INCLUDING CONTENTS` est implicite

⇒ les segments d'annulation stockés dans le tablespace sont supprimés.

Par contre, la clause doit être mentionnée avec l'option `AND DATAFILES` pour supprimer les fichiers de données associés.

19.6 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les tablespaces d'annulation et les segments d'annulation :

- `V$TABLESPACE` ou `DBA_TABLESPACES` : informations sur les tablespaces (dont les tablespaces d'annulation)
- `DBA_DATA_FILE` ou `V$DATAFILE` : informations sur les fichiers
- `DBA_FREE_SPACE` : espace disponible
- `DBA_SEGMENTS` : liste de segments
- `DBA_UNDO_EXTENTS` : liste des extents alloués dans le tablespace d'annulation
- `DBA_ROLLBACK_SEGS` : informations sur les segments d'annulation et/ou les rollback segments
- `V$ROLLNAME` : liste des segments d'annulation et/ou des rollback segments actuellement `ONLINE` ou `PENDING OFFLINE`



- V\$ROLLSTAT : statistiques sur les segments d'annulation et/ou les rollback segments actuellement ONLINE ou PENDING OFFLINE
- V\$UNDOSTAT : statistiques sur les informations d'annulation
- V\$TRANSACTION : informations sur les transactions actives

La vue V\$UNDOSTAT donne des statistiques sur les informations d'annulation générées sur les dernières 24 heures. La collecte est effectuée par plages de 10 minutes ; la vue contient donc 144 lignes, une pour chaque plage de 10 minutes sur les dernières 24 heures.

La première ligne de la vue correspond à la plage en cours (peut faire moins de 10 minutes). Les colonnes intéressantes de la vue sont les suivantes :

- ⇒ BEGIN_TIME : Date/heure de début de la plage
- ⇒ END_TIME : Date/heure de fin de la plage
- ⇒ UNDOBLKS : Nombre de blocs d'annulation utilisés en cumulé sur la période
- ⇒ TXNCOUNT : Nombre de transactions totales sur la période
- ⇒ MAXQUERYLEN : Durée en seconde de la requête la plus longue sur la période
- ⇒ MAXCONCURRENCY : Nombre maximum de transactions simultanées sur la période

La vue V\$ROLLSTAT est utilisée pour l'optimisation des rollback segments.

Dans la pratique il faut surveiller le nombre de fois où le rollback segment a dû s'étendre (EXTENDS) par rapport au nombre de fois où il a pu tourner sans problème (WRAPS)

Si les rollback segments sont bien dimensionnés, le ratio EXTENDS/WRAPS est faible.

- ⇒ Un ratio supérieur à 10% indique que les rollback segments sont trop petits.

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les rollback segments :

- DBA_ROLLBACK_SEGS : informations sur tous les rollback segments
- V\$ROLLNAME : liste des rollback segments ONLINE
- V\$ROLLSTAT : statistiques sur les rollback segments ONLINE
- V\$TRANSACTION : informations sur les transactions actives

V\$ROLLNAME	
USN	Numéro du segment
NAME	Nom du segment d'annulation ou de rollback



V\$TRANSACTION

SES_ADDR	Adresse de la session utilisateur qui exécute la transaction une jointure avec V\$SESSION permet de récupérer des informations sur la session
START_TIME	Date et heure de démarrage de la transaction (sous forme de chaîne)
XIDUSN	Numéro du segment d'annulation ou de rollback utilisé pour la transaction

V\$UNDOSTAT

BEGIN_TIME	Date et heure de démarrage de la plage
END_TIME	Date et heure de fin de plage
UNDOBLKS	Nombre de blocs d'annulation utilisés en cumulé sur la période
TXNCOUNT	Nombre de transactions totales sur la période
MAXQUERYLEN	Durée en seconde de la requête la plus longue sur la période
MAXCONCURRENCY	Nombre maximum de transactions simultanées sur la période

V\$ROLLSTAT

USN	Numéro du rollback segment
EXTENT	Nombre d'extents dans le segment d'annulation ou de rollback
RSSIZE	Taille actuelle utile en octet (sans le bloc d'en-tête) du segment d'annulation ou de rollback
TABLESPACE_NAME	Nom du tablespace
WRITES	Nombre d'octets écrits dans le segment
OPTSIZE	Valeur OPTIMALE en octet du segment d'annulation ou de rollback
HWMSIZE	Plus grande taille jamais atteinte en octet par le segment
SHRINKS	Nombre de fois où le segment a rétréci
WRAPS	Nombre de fois où le segment a tourné (où il est venu réécrire dans le premier extent sans avoir eu besoin de s'étendre)
EXTENDS	Nombre de fois où le segment s'est étendu (allocation d'un nouvel extent)
AVESHRINK	Taille moyenne en octets des rétrécissements du segment d'annulation ou de rollback.



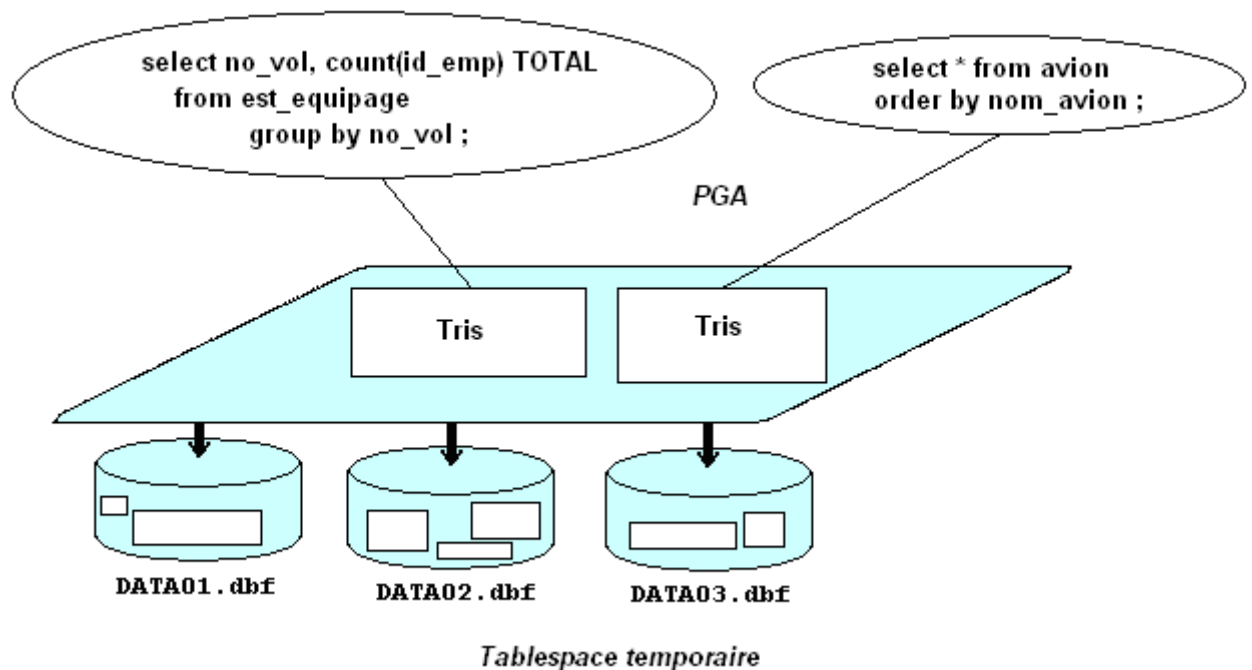
20 Tablespaces temporaires

Le tablespace temporaire héberge les segments temporaires issus des requêtes ou de commandes SQL telles que :

- ♦ select ... order by
- ♦ select ... group by
- ♦ select distinct ...
- ♦ les requêtes ensemblistes (UNION, MINUS, INTERSECT)
- ♦ create INDEX ...
- ♦ gestion des statistiques (DBMS_STATS)
- ♦ les jointures par tri-fusion (SORT, MERGE, JOIN)
- ♦ les tables temporaires (CREATE GLOBAL TEMPORARY TABLE ...)

Lorsqu'une requête nécessite un tri, Oracle tente de faire le tri en mémoire dans la PGA du processus serveur qui exécute la requête. S'il ne peut pas car le tri ne tient pas en mémoire, Oracle le découpe en morceaux et trie chaque morceau en stockant des résultats intermédiaires sur disque dans des segments temporaires.

Au bout d'un certain temps ces segments temporaires sont supprimés par Oracle.





Du point de vue des performances, il est conseillé de dédier un tablespace aux segments temporaires en utilisant le mot clé `TEMPORARY` à la création du tablespace. obligatoire depuis la 9i.

- Evite la fragmentation et optimise les performances !

Exemple

Si 50 users font un tri de 1 Méga chacun, on aura 50 PGA de 1 Méga et si il n'y a pas assez de place mémoire alors oracle utilisera le tablespace temporaire.

Les segments doivent avoir une taille correspondant à un multiple de `SORT_AREA_SIZE` (5 à 10 fois) + un bloc d'en-tête pour les données de contrôle.

Depuis Oracle9i, le paramètre `PGA_AGGREGATE_TARGET` permet la gestion dynamique de la PGA et des paramètres `SORT_AREA_SIZE` et autres. Depuis la version 9i il est également possible de définir un tablespace temporaire par défaut, dès la création de la base de données.

Il est attribué par défaut aux utilisateurs qui ont été créé sans tablespace temporaires par défaut (si la clause `TEMPORARY TABLESPACE` n'est pas précisée dans l'ordre `SQL CREATE USER`).



Le tablespace temporaire par défaut ne peut pas être supprimé. Si possible, mettre le tablespace temporaire sur un disque à part.

Le tablespace temporaire par défaut ainsi créé est forcément géré localement. Il a une taille de bloc correspondant au paramètre `DB_BLOCK_SIZE`.

- informations sur les tablespaces par défaut

```
SQL> col property_value for A15
SQL> select property_name, property_value
2  from database_properties
3  where property_name like 'DEFAULT_%'
4  /
```

PROPERTY_NAME	PROPERTY_VALUE
DEFAULT_TEMP_TABLESPACE	TEMP
DEFAULT_PERMANENT_TABLESPACE	USERS
DEFAULT_TBS_TYPE	SMALLFILE



20.1 Créer un tablespace temporaire

Depuis Oracle 8i, un tablespace temporaire peut être géré localement. Dans ce cas, il utilise des fichiers temporaires.

Depuis la version 9i, il est possible de définir un tablespace temporaire par défaut à la création de la base de données afin de recueillir les tris des requêtes de tous les utilisateurs de la base de données.

Il est créé avec l'ordre SQL `CREATE TEMPORARY TABLESPACE`.

```
CREATE [ BIGFILE | SMALLFILE ] TEMPORARY TABLESPACE Ts_nom
TEMPFILE spécification_fichier_temp [,...]
[EXTENT MANAGEMENT LOCAL] [UNIFORM [SIZE valeur [K|M] ] ]
[ TABLESPACE GROUP nom_group ]
;
```

♦ spécification_fichier_temp

```
'nom_fichier' [ SIZE valeur [K|M|G|T] ] [REUSE]
AUTOEXTEND { OFF|ON [ NEXT valeur [K|M|M|T] ]
[ MAXSIZE { UNLIMITED | valeur [K|M|G|T] } ] }
```

- par défaut la taille des extents est de 1Mo, la clause `SIZE` peut être utilisée pour spécifier une autre taille, dans ce cas le mot clé `UNIFORM` doit être mentionné.
- `BIGFILE`, précise que le fichier est gros mais dans ce cas un seul fichier pourra être rattaché au tablespace.
- `TABLESPACE GROUP`, spécifie le groupe d'appartenance du tablespace. Si le groupe n'existe pas, il est créé implicitement. Par défaut, le tablespace n'appartient à aucun groupe.



Les fichiers de données d'un tablespace temporaire géré localement sont particuliers : Ils sont dits temporaires
« `TEMPFILE` ».

```
SQL> create temporary tablespace ts_essai
2 tempfile 'C:\app\guest\oradata\BORA\essai_temp.dbf'
3 size 5M;

Tablespace created.
```

20.2 Groupes de tablespaces temporaires

Avant la version 10g, une requête ne pouvait utiliser qu'un seul tablespace temporaire. Ceci posait des problèmes de performance pour les requêtes exécutées en parallèle (chaque processus serveur sollicite un accès au tablespace temporaire, d'où des problèmes de contentions et de performances).

Depuis la version Oracle 10g, il est possible de définir des groupes de tablespaces temporaires. Le nom d'un groupe peut être utilisé partout où un nom de tablespace temporaire est utilisé.



Ils respectent les règles suivantes :

- ⇒ Un groupe ne peut pas porter le même nom qu'un tablespace temporaire.
- ⇒ Un groupe est créé lorsqu'un tablespace est affecté au groupe.
- ⇒ Un groupe est implicitement supprimé lorsque le dernier tablespace est retiré du groupe.

Vous pouvez changer le groupe d'appartenance du tablespace par la commande `ALTER TABLESPACE` en précisant le nouveau nom du groupe auquel il appartient, ou en spécifiant une chaîne vide, ce qui le retire du groupe.

```
ALTER TABLESPACE nom tablespace group nom  
;
```

```
SQL> select * from dba_tablespace_groups ;  
no rows selected  
SQL> alter tablespace ts_essai tablespace group group_essai  
2 /  
  
Tablespace altered.  
SQL> select * from dba_tablespace_groups;  
GROUP_NAME          TABLESPACE_NAME  
-----  
GROUP_ESSAI          TS_ESSAI  
  
SQL> alter tablespace ts_essai tablespace group « ;  
Tablespace altered.  
SQL> select * from dba_tablespace_groups;  
no rows selected
```

20.3 Administrer les tablespaces temporaires

L'administration d'un tablespace temporaire géré localement s'effectue avec les ordres SQL habituels, en remplaçant le mot clé `DATAFILE` par le mot clé `TEMPFILE` :

- ♦ `ALTER TABLESPACE`
- ♦ `ALTER DATABASE` pour la gestion des fichiers de données
- ♦ `DROP TABLESPACE` pour la suppression

Il existe néanmoins quelques restrictions :

- ⇒ Un tablespace temporaire ne peut pas être mis `OFFLINE`.
- ⇒ Les fichiers d'un tablespace temporaire ne peuvent pas être déplacés, ils doivent être supprimés puis recréés.
- ⇒ Toujours en mode `NOLOGGING`, les modifications ne sont pas enregistrées dans les fichiers de Redo Log (intéressant pour les performances).
- ⇒ Il ne peut pas être sauvegardé.



20.3.1 Agrandir un tablespace temporaire

Il est possible d'ajouter un fichier temporaire à un tablespace temporaire.

```
ALTER TABLESPACE nom_ts_temp
ADD TEMPFILE spécification_fichier_temp
;
```

```
SQL> alter tablespace ts_essai
2 add tempfile 'D:\Oracle\product\10.1.0\ORADATA\ORCL\essai_temp02.dbf'
3 size 5M;
```

Tablespace altered.

```
SQL> select t.name ts, bigfile, enabled, f.name
2 from v$tablespace t, v$tempfile f
3 where t.ts#=f.ts#
4 UNION
5 select t.name ts, bigfile, enabled, f.name
6 from v$tablespace t, v$datafile f
7 where t.ts#=f.ts#
8 /
```

TS	BIG	ENABLED	NAME
EXAMPLE	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\EXAMPLE01.DBF
SYSAUX	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\SYSAUX01.DBF
SYSTEM	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\SYSTEM01.DBF
TEMP	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\TEMP01.DBF
TS_ESSAI	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP.DBF
TS_ESSAI	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP02.DBF
UNDOTBS1	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\UNDOTBS01.DBF
USERS	NO	READ WRITE	D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\USERS01.DBF

8 rows selected.

20.3.2 Modifier la clause AUTOEXTEND :

```
ALTER DATABASE TEMPFILE 'nom complet' [, ...] clause autoextend
;
```

```
SQL> alter database
2 tempfile 'D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP02.DBF'
3 autoextend on
4 /
```

Database altered.

```
SQL> select file_name, autoextensible, maxbytes, maxblocks
2 from dba_temp_files;
```



```
SQL> select file_name, autoextensible, maxbytes, maxblocks
2 from dba_temp_files;

FILE_NAME                                     AUT      MAXBYTES      MAXBLOCKS
-----
D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\TEMP01.DBF  YES 3,4360E+10      4194302
D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP.DBF NO          0          0
D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP02.DBF YES 3,4360E+10      4194302
```

20.3.3 Modifier la taille d'un fichier temporaire

```
ALTER DATABASE TEMPFILE 'nom complet' [, ...] RESIZE valeur [L|M]
;
```

```
SQL> alter database
2 tempfile 'D:\ORACLE\PRODUCT\10.1.0\ORADATA\ORCL\ESSAI_TEMP02.DBF'
3 resize 2M
4 /

Base de données modifiée.
```

20.3.4 Rétrécir un tablespace temporaire (Nouveautés 11g)

Certaines opérations d'administration peuvent demander beaucoup d'espace temporaire et donc d'avoir un tablespace temporaire de très grande taille.

Pour faire face à ce type de besoin, la version 11g propose la commande `SHRINK` qui permet de réduire la taille du tablespace ou de l'un de ses fichiers :

```
ALTER TABLESPACE nom tbs SHRINK SPACE [ KEEP taille [K|M|G|T] ] ;
ALTER TABLESPACE nom tbs SHRINK TEMPFILE '/chemin/nom fichier.dbf' | No fichier
[ KEEP taille [K|M|G|T] ] ;
```

- Cette commande permet de réduire la taille du tablespace temporaire ou la taille d'un de ses fichiers.
- La clause `KEEP` permet de préciser une taille minimum à conserver pour le tablespace ou le fichier

Exemples

```
ALTER TABLESPACE temp SHRINK SPACE ;
ALTER TABLESPACE temp SHRINK TEMPFILE '/oracle/oradata/orcl/temp01.dbf' ;
```

Il est possible de préciser une quantité d'espace libre supplémentaire laissée dans le tablespace en utilisant la clause `KEEP` pour le tablespace ou le fichier temporaire, mais si la clause `KEEP` est trop basse une erreur est retournée.

Sans clause `KEEP` Oracle tente de libérer le maximum d'espace au niveau du tablespace (de tous les fichiers rattachés) ou du fichier spécifié.



Exemples

```
ALTER TABLESPACE temp SHRINK SPACE KEEP 100M ;  
ALTER TABLESPACE temp SHRINK  
TEMPFILE '/oracle/oradata/orcl/temp01.dbf'  
KEEP 100M ;
```

20.3.5 Supprimer un tablespace temporaire

S'effectue avec l'ordre SQL DROP TABLESPACE.

```
DROP TABLESPACE nom [ INCLUDING CONTENTS [ AND DATAFILES ] ]  
;
```

```
SQL> drop tablespace ts_essai including contents and datafiles  
2 /  
Tablespace dropped.
```

20.4 Définir un tablespace temporaire par défaut

La création du tablespace temporaire par défaut peut se faire après création de la base de données avec l'ordre CREATE TEMPORARY TABLESPACE.

Il faut ensuite désigner ce tablespace temporaire comme tablespace temporaire par défaut en utilisant la commande ci-dessous.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE nom tbs  
;
```

Dès l'affectation du nouveau tablespace temporaire comme tablespace par défaut, Oracle mets les segments de temporaire dans le nouveau tablespace et libère l'ancien tablespace.

Il est possible de définir un groupe de tablespace temporaire par défaut à la place d'un tablespace temporaire par défaut.



En cas de perte de fichier temporaire supplémentaire, oracle ouvre la base normalement mais ne le signale pas directement. Par contre une erreur est signalée dans le fichier de trace et le fichier des alertes : attention aux restaurations !



20.5 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les tablespaces et les fichiers de données :

- `DBA_TABLESPACES` : informations sur les tablespaces
- `V$TABLESPACE` : informations sur les tablespaces (à partir du fichier de contrôle)
- `DBA_TABLESPACE_GROUPS` : informations sur les groupes de tablespaces.
- `DBA_TEMP_FILES` : informations sur les fichiers de données des tablespaces temporaires gérés localement.
- `V$TEMPFILE` : informations sur les fichiers de données des tablespaces temporaires gérés localement.
- `V$SORT_SEGMENT` : supervision du stockage des segments temporaires.
- `DATABASE_PROPERTIES` : informations sur les propriétés par défaut de la base de données, pour le tablespace temporaire par défaut, le nom de la propriété est `DEFAULT_TEMP_TABLESPACE`.
- **`DBA_TEMP_FREE_SPACE` : nouvelle vue en version 11g**, qui permet de gérer plus facilement les tablespaces temporaires. Elle contient la taille totale utilisée pour le tablespace temporaire, ainsi que l'espace temporaire utilisé et l'espace temporaire disponible.

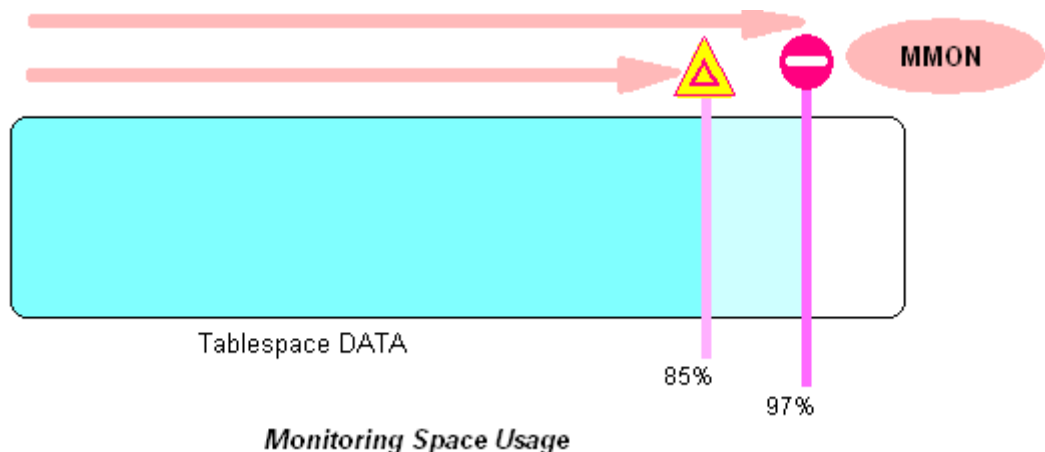
```
Select * from dba_temp_free_space ;
```

TABLESPACE_NAME	TABLESPACE_SIZE	ALLOCATED_SPACE	FREE_SPACE
TEMP	18,939,904	4,259,840	14,680,064



21 Monitoring de l'utilisation d'un tablespace

Les seuils du tablespace sont définis comme des pourcentages par rapport à la taille du tablespace. Quand le taux de remplissage du tablespace arrive à une de ces 2 limites (voir schéma), une alerte appropriée est déclenchée ou inhibée.



Les valeurs des seuils critiques et d'avertissement peuvent être configurées.

Si une valeur de seuil n'est pas spécifiée, les comportements suivants seront appliqués :

- ⇒ Les valeurs par défaut sont 85% de taux de remplissage pour l'alerte d'avertissement et 97% pour le seuil critique.
- ⇒ Les alertes sont désactivées par défaut, les valeurs de seuil sont donc à NULL. Ces valeurs peuvent être redéfinies par la suite, à tout moment.

Le process d'arrière plan `MMON` vérifie toutes les 10 minutes ces seuils d'alertes (dépassés ou redescendu). Une alerte est déclenchée dans chacun de ces cas (dépassé ou redescendu)

- ◆ Les alertes ne doivent pas être actives pour les tablespaces en lecture seule (Read-only) ou bien pour les tablespaces `OFFLINE` car ça n'a aucun sens.
- ◆ Dans les tablespaces temporaires, les valeurs de seuil doivent être définies comme une limite de l'espace utilisé dans le tablespace par les sessions.
- ◆ Pour le tablespace `UNDO`, un `EXTENT` est réutilisable s'il ne contient pas des segments `UNDO` actifs. Pour le calcul de la violation des seuils, la somme des `Extents` actifs est considérée comme espace utilisé.
- ◆ Pour les tablespaces avec des fichiers auto-extensibles, les seuils sont calculés en fonction de la taille maximale spécifiée du fichier ou de la taille maximale supportée par l'OS.

Le suivi de ces opérations se fait en utilisant les tables `DBA_THRESHOLDS` qui contient la configuration des alertes et `DBA_ALERT_HISTORY` qui contient l'historique des alertes.



```
-- visualiser les alertes min et max définies sur les tablespaces
--

select warning_value,critical_value
from dba_thresholds
where metrics_name='Tablespace Space Usage' and
object_name is null
/

select warning_value,critical_value
from dba_thresholds
where metrics_name='Tablespace Space Usage' and
object_name='DATA'
/

select reason,resolution
from dba_alert_history
where object_name='DATA'
/
```

21.1 Configuration des seuils de tablespace

Vous pouvez utiliser le package DBMS_SEVR_ALERT pour définir vos propres valeurs par défaut pour l'utilisation de l'espace dans le tablespace. Les procédures SET_THRESHOLD, et GET_THRESHOLD vous permettent de le faire.

- définit un seuil d'avertissement à 80% du taux de remplissage
- et un seuil critique à 95% pour le tablespace DATA.
- Dès que le pourcentage de remplissage est égal ou supérieur à 80%
- une alerte d'avertissement est déclenchée.
- L'alerte critique est déclenchée quand le taux de remplissage est égal ou supérieur à 95%;

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD
(
Dbms_server_alert.tablespace_pct_full ,
Dbms_server_alert.operator_ge, 80 ,
Dbms_server_alert.operator_ge, 95, 1, 1, NULL ,
Dbms_server_alert.object_type_tablespace, 'DATA'
) ;
```

- remet les valeurs par défaut pour le tablespace appelé DATA.

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD
(
Dbms_server_alert.tablespace_pct_full ,
NULL, NULL, NULL, NULL, 1, 1, NULL,
Dbms_server_alert.object_type_tablespace, 'DATA'
) ;
```

- désactiver le mécanisme de traçage
- de l'utilisation de l'espace pour le tablespace DATA

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD
(
Dbms_server_alert.tablespace_pct_full ,
Dbms_server_alert.operator_do_not_check, '0' ,
Dbms_server_alert.operator_do_not_check, '0' , 1, 1, NULL ,
Dbms_server_alert.object_type_tablespace, 'DATA'
) ;
```

- définir vos propres valeurs de seuil par défaut
- elles deviennent la valeur de mesure « Tablespace Space Usage »

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD
(
Dbms_server_alert.tablespace_pct_full ,
Dbms_server_alert.operator_ge, 80 ,
Dbms_server_alert.operator_ge, 95, 1, 1, NULL ,
```



```
Dbms_server_alert.object_type_tablespace, NULL  
    ) ;
```

- revenir aux valeurs de 85% et 97% comme valeur de seuil par défaut

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD
```

```
(  
Dbms_server_alert.tablespace_pct_full ,  
NULL, NULL, NULL, NULL, 1, 1, NULL,  
Dbms_server_alert.object_type_tablespace, NULL  
    ) ;
```

- utilisée pour lire les valeurs et les seuils
- pour l'ensemble de la base de données

```
EXECUTE DBMS_SERVER_ALERT.GET_THRESHOLD
```

```
(  
Dbms_server_alert.tablespace_pct_full ,  
:wo, :wv, :co, :cv, :op, :cocc, NULL,  
Dbms_server_alert.object_type_tablespace, NULL  
    ) ;
```

- utilisée pour lire les valeurs et les seuils
- pour le tablespace DATA

```
EXECUTE DBMS_SERVER_ALERT.GET_THRESHOLD
```

```
(  
Dbms_server_alert.tablespace_pct_full ,  
:wo, :wv, :co, :cv, :op, :cocc, NULL,  
Dbms_server_alert.object_type_tablespace, 'DATA'  
    ) ;
```



22 Mémoire dynamique et performances

En version 10g d'Oracle, *Automatic PGA Memory Management* est activée par défaut, à moins que `PGA_AGGREGATE_TARGET` soit explicitement fixée à zéro ou `WORKAREA_SIZE_POLICY` soit explicitement posée à `MANUAL`.

`PGA_AGGREGATE_TARGET` est par défaut à 20% de la taille de la `SGA`.

Automatic PGA Memory Management permet de conseiller l'utilisateur de façon automatique sur les problèmes de performances de la base de données et de proposer des solutions adaptées.

Avec Oracle 9i, était apparue la notion de *granule*, ainsi que la possibilité de redimensionner la mémoire de façon interactive.

Elle peut toujours être modifiée dynamiquement alors que l'instance est en cours de fonctionnement :

- ⇒ Augmentée ou diminuée
- ⇒ Sans devoir arrêter la base
- ⇒ En modifiant la valeur des paramètres qui la dimensionnent, par l'intermédiaire de l'ordre SQL `ALTER SYSTEM`

En cas d'augmentation, la taille maximum de la `SGA` est définie par le paramètre `SGA_MAX_SIZE`, qui n'est pas dynamique et calculé par défaut si non spécifié.



La taille de la `SGA` peut augmenter ou diminuer en fonction des besoins internes d'Oracle. C'est notamment le cas au démarrage où les valeurs des différents paramètres de dimensionnement de la `SGA` peuvent être adaptés par Oracle.

22.1 La notion de granule

Un granule est une quantité de mémoire qui peut être allouée à une structure de la `SGA` :

- ⇒ D'une taille de 4 Mo si la taille totale de la `SGA` est inférieure à 128 Mo
- ⇒ D'une taille de 16 Mo autrement

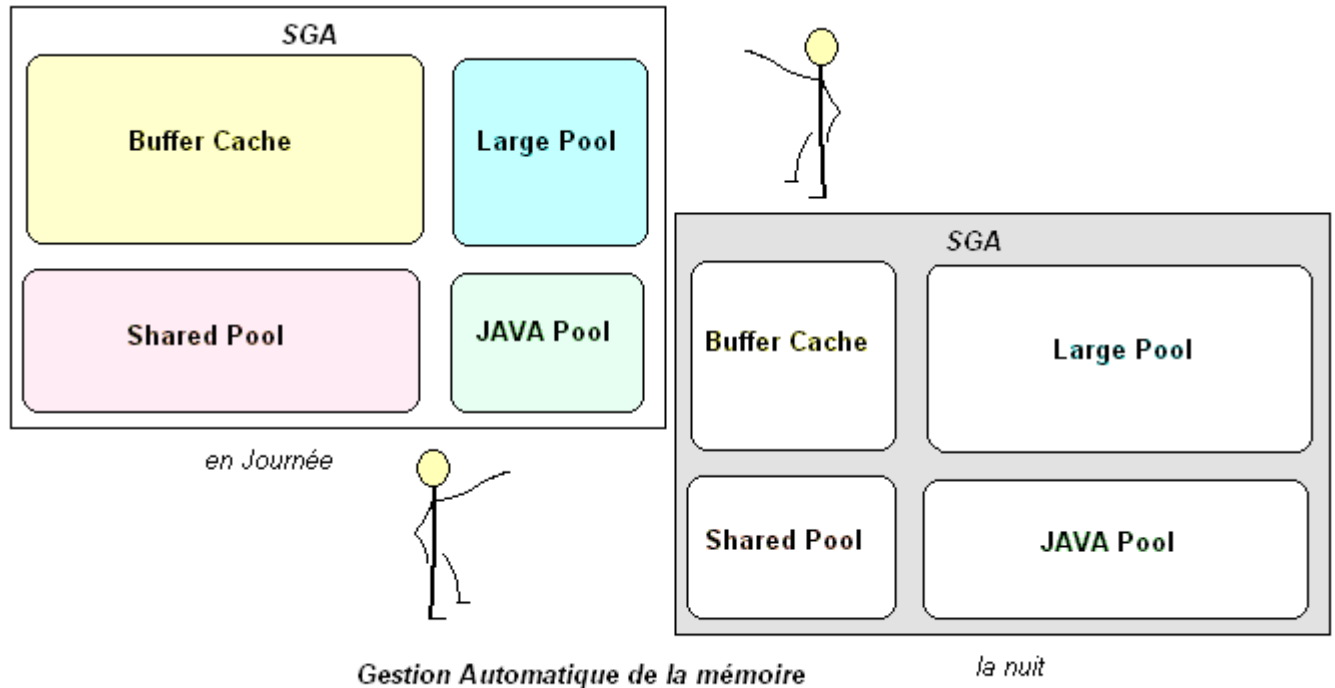
La vue `V$SGA_COMPONENT` permet de visualiser la taille du granule d'une base de données.

22.2 Gestion automatique du partage de la mémoire en 10g

La gestion automatique du partage de la mémoire est une amélioration importante d'autogestion d'Oracle 10g.

Cette fonctionnalité automatise la gestion des structures les plus importantes de mémoire partagée utilisée par toutes les instances d'une base de données Oracle.





Avec « *Automatic Shared Memory Management* » quand les jobs OLTP s'exécutent, le *Buffer Cache* prend la plus grande partie de la mémoire afin d'avoir une bonne performance d'entrée/sortie.

Par contre, lorsque le job batch démarrera plus tard, la mémoire sera automatiquement allouée vers le *Large Pool* afin d'être utilisée par des opérations parallèles de requêtes en évitant les erreurs de débordement de mémoire (*Memory overflow*).

22.2.1 Principes de tuning de la SGA

L'outil « *Automatic Shared Memory Management* » utilise un nouveau processus d'arrière plan appelé *Memory Manager* (MMAN).

MMAN agit comme un distributeur (*broker*) de mémoire et coordonne la taille allouée aux différents composants de la SGA. Il conserve une trace de la taille des composants et des opérations de dimensionnement en attente.

MMAN observe le système et le *Workload* (travail en cours) afin de déterminer la distribution de mémoire optimale. Il exécute ces vérifications toutes les « *quelques minutes* » afin que la mémoire soit toujours affectée là où on en a besoin.

En absence de gestion automatique du partage mémoire les composants doivent être dimensionnés afin d'anticiper leurs besoins en mémoire.

Basé sur les informations extraites du *Workload*, MMAN exécute les actions suivantes :

- ⇒ Effectue des statistiques périodiques dans l'arrière plan.
- ⇒ Utilise divers outils de conseils sur la mémoire.
- ⇒ Exécute des analyses de type « *What-IF* » (SI) pour déterminer la meilleure distribution de la mémoire possible.
- ⇒ Affecte de la mémoire là où elle en a besoin.
- ⇒ Sauvegarde la taille des composants pendant les arrêts de la base (*shutdown*) si un *SPFILE* est utilisé. (La taille peut être récupérée à partir des valeurs existantes avant l'arrêt).



La mémoire est redistribuée entre les composants suivants :

- ⇒ DB_CACHE_SIZE
- ⇒ SHARED_POOL_SIZE
- ⇒ LARGE_POOL_SIZE
- ⇒ JAVA_POOL_SIZE

Avant la version 10g, la mémoire additionnelle était allouée à la SGA fixe et la mémoire additionnelle était entre 10 et 20Mo.

Le paramètre SGA_TARGET inclut toute la mémoire dans la SGA, comprenant aussi les composants dimensionnés automatiquement ou manuellement ainsi que toutes les allocations internes faites pendant le démarrage.

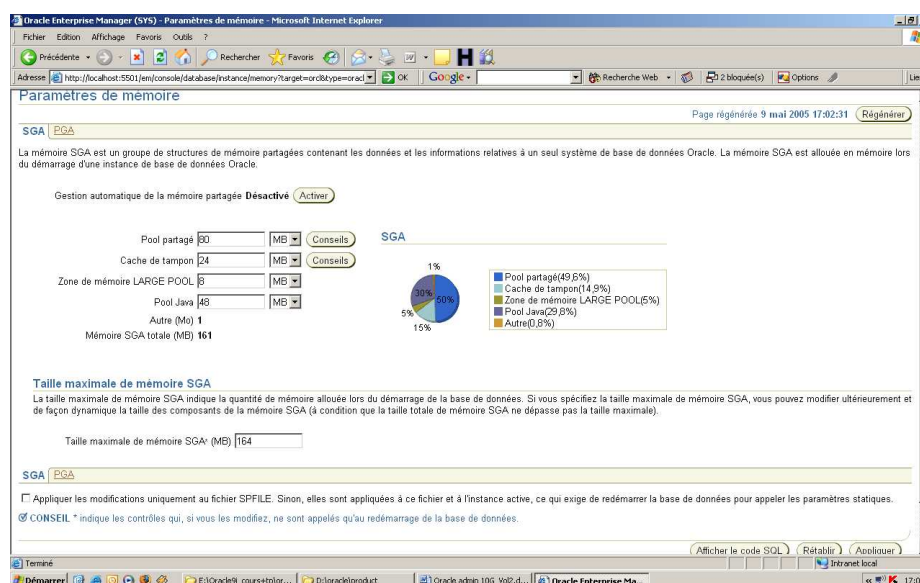


Mettre SGA_TARGET à la valeur SGA_MAX_SIZE !

22.2.2 SGA_TARGET et le Database Control (OEM)

Vous pouvez utiliser le Database Control pour configurer le « *Automatic shared Memory Management* » comme indiqué dans le mode opératoire ci-dessous (dans l'ordre) :

- ◆ Cliquez sur l'onglet « Administration »
- ◆ Sélectionner sous le nom de l'instance « Memory Parameter »
- ◆ Cliquez l'onglet « SGA »
- ◆ Cliquez sur le bouton « ENABLE » pour « *Automatic shared Memory Management* » puis saisir la taille totale de la SGA (en Mo). Cette valeur est affectée à SGA_TARGET.



22.2.3 Configuration manuelle de SGA_TARGET

« *Automatic shared Memory Management* » se configure avec le paramètre `SGA_TARGET`.

Si vous spécifiez une valeur différente de zéro, les 4 pools de mémoire seront dimensionnés automatiquement :

- ♦ Database buffer cache (pool par défaut)
- ♦ Shared pool (pool partagé)
- ♦ Large pool
- ♦ Java pool

Si `SGA_TARGET` = zéro (valeur par défaut), « *Automatic shared Memory Management* » est inactivé.

Les paramètres individuels utilisés dans les versions antérieures pour spécifier les tailles des composants dimensionnés automatiquement sont toujours actifs.

Les paramètres d'initialisation qui dimensionnent ces pools sont maintenant appelés paramètres « *Auto-tuned* » de la SGA.

Les buffers suivants sont des composants dimensionnés manuellement :

- ♦ Log buffer
- ♦ Autres buffer cache (`KEEP/RECYCLE`, autres dimensions de blocks associés au buffer cache).
- ♦ Streams pool
- ♦ SGA fixe et autre allocation interne



En version 10g et 11g le paramètre `STATISTICS_LEVEL` doit être positionné à la valeur `TYPICAL` !

22.2.4 Comportement des paramètres Auto-tuned

Quand `SGA_TARGET` n'est pas définie ou est égale à zéro les paramètres auto-réglés de la SGA se comportent comme dans les versions précédentes,

⇒ sauf `SHARED_POOL_SIZE` qui fait exception.

Les allocations des *overhead* internes (temps utilisé pour le système lui-même) pour les méta-données (comme les structures de données pour les processus, les sessions, etc..) sont maintenant incluses dans la valeur du paramètre `SHARED_POOL_SIZE`.

Par conséquent, vous pouvez avoir besoin d'augmenter la valeur du paramètre `SHARED_POOL_SIZE` pour migrer vers Oracle 10g afin de prendre en compte ces allocations.

Par exemple, si vous utilisiez une valeur de 256 Mo dans une version précédente et la valeur des allocations internes était de 32 Mo en Oracle 10g, vous devrez mettre la valeur de `SHARED_POOL_SIZE` à 288 Mo pour obtenir la même taille de pool.



- calcule de la valeur totale de la Shared Pool
- en incluant cet overhead (supplémentaire) interne.

```
SELECT SUM(bytes)/1024/1024
FROM   V$SGASTAT
WHERE  pool = 'shared pool'
/
```

Cette requête permet de déterminer la nouvelle valeur de la *Shared Pool* en version 10g.

Quand `SGA_TARGET` est différent de 0, les paramètres auto-réglés de la *SGA* ont une valeur par défaut égale à zéro. Ces composants sont dimensionnés par l'algorithme du « *Automatic shared Memory Management* ».

Si leur valeur est différente de 0, ces valeurs sont utilisées comme une limite inférieure par l'algorithme d'autoréglage.

Par exemple si `SGA_TARGET` = 8 Go et `SHARED_POOL_SIZE` = 1Go, ceci indique à l'algorithme du « *Automatic Shared Memory Management* » de ne jamais baisser la valeur de la *Shared Pool* en dessous de 1Go et seules des valeurs supérieures seront acceptées.

Pour déterminer la taille effective des composants auto-réglés de la *SGA*, utilisez la requête suivante :

```
SELECT component, current_size/1024/1024
FROM   V$SGA_DYNAMIC_COMPONENTS
/
```

22.2.5 Comportement des paramètres Manuels

Les paramètres manuels de la *SGA* sont :

- ⇒ `DB_KEEP_CACHE_SIZE`
- ⇒ `DB_RECYCLE_CACHE_SIZE`
- ⇒ `DBn_CACHE_SIZE` (n= 2,4,8,16,32)
- ⇒ `LOG_BUFFER`
- ⇒ `STREAMS_POOL_SIZE`

Ils sont spécifiés par l'utilisateur et leur taille détermine la taille de leur composant.

Quand `SGA_TARGET` a une valeur, la taille totale des paramètres manuels est soustraite à cette valeur. Le reste donne la valeur des composants auto-réglés de la *SGA*.

Par exemple si `SGA_TARGET` = 8 Go et `DB_KEEP_CACHE_SIZE` = 1Go alors la taille totale des 4 composants auto-réglés est limitée à 7 Go.

Les 7 Go incluent la *SGA* fixe et le Buffer Log et seulement après avoir alloué ceci, le reste de la mémoire est divisée entre les composants auto-réglés. La taille du `KEEP_CACHE` est de 1 Go comme spécifié par le paramètre.

Vues V\$PARAMETER

Quand vous spécifiez une valeur différente de zéro pour `SGA_TARGET` et que vous ne spécifiez pas de valeur pour les paramètres auto-réglés, la valeur de ces paramètres dans la vue `V$PARAMETER` est 0 et la valeur de la colonne `ISDEFAULT` = `TRUE`.

Si vous avez spécifié une valeur pour au moins un des paramètres auto-réglés, la valeur affichée dans la vue `V$PARAMETER` est la valeur que vous avez spécifiée pour ce paramètre.



```
SELECT name, value, isdefault  
FROM V$PARAMETER  
WHERE name like '%size'  
/
```

22.2.6 Redimensionner SGA_TARGET

Le paramètre dynamique `SGA_TARGET` est :

- ◆ Dynamique
- ◆ Peut être réduit jusqu'à la valeur `SGA_MAX_SIZE`
- ◆ Peut être augmenté jusqu'à ce que tous les composants *auto_tuned* aient atteint leur taille minimale.

La modification du paramètre `SGA_TARGET` influence automatiquement la taille des autres composants. Cette modification peut se faire *via* le *Database Control* ou une commande `ALTER SYSTEM`.

22.2.7 Désactiver la gestion automatique de la mémoire en version 10g

Il est possible de désactiver le partage de gestion automatique de la mémoire en mettant `SGA_TARGET` à zéro, dans ce cas les valeurs de tous les paramètres auto-dimensionnés prennent la taille définie en mémoire des composants correspondants, même si les utilisateurs ont spécifié des valeurs non nulles pour ces paramètres.

22.3 Gestion automatique de la mémoire en 11g

La gestion automatique du partage de la mémoire reste une amélioration importante d'autogestion d'Oracle 11g. Elle permet une gestion automatique de l'ensemble de la mémoire (SGA et PGA).

⇒ Le paramètre `MEMORY_TARGET` permet d'activer cette fonctionnalité.

Ce paramètre indique la taille globale de la SGA et de la PGA utilisée par Oracle.

Les tailles respectives de ces zones peuvent changer en fonction des besoins.

La taille globale que la totalité de la mémoire Oracle peut atteindre est donnée par le paramètre :

⇒ `MEMORY_MAX_TARGET`

`MEMORY_MAX_TARGET` est un paramètre statique alors que `MEMORY_TARGET` est un paramètre dynamique.

Le processus d'arrière plan *Automatic Memory Management* utilise et adapte automatiquement les paramètres `SGA_TARGET` et `PGA_TARGET` pour allouer la mémoire à l'instance.

Le paramètre `MEMORY_MAX_TARGET` définit une valeur maximum de mémoire utilisée par Oracle et la bloque. Il doit obligatoirement avoir une valeur supérieure ou égale à la valeur de `MEMORY_TARGET`.



22.3.1 Désactiver la gestion automatique de la mémoire en version 11g

Si ces deux paramètres sont positionnés à zéro, la gestion de la mémoire dynamique est désactivée. Dans ce cas, les valeurs des paramètres `SGA_TARGET` et `PGA_AGGREGATE_TARGET` sont désactivés (valeur à zéro), et il faut utiliser les anciens paramètres `DB_CACHE_SIZE`, `SHARED_POOL_SIZE`, etc..

22.3.2 La vue dynamique V\$MEMORY_TARGET_ADVICE

Cette vue dynamique de performances, permet de suivre l'allocation dynamique et visualiser les différentes valeurs de l'allocation dynamique de la mémoire.

Cette vue contient les colonnes :

- ♦ Memory size : taille réelle de la mémoire totale allouée à l'instance
- ♦ Size_factor : coefficient de taille
- ♦ Estd_db_time : taille de l'instance utilisée en mémoire en moyenne par rapport aux facteurs size-factor et time_factor.
- ♦ Time_factor : coefficient de temps
- ♦ Version :

```
|Select * from v$memory_target_advice ;
```

La vue `V$MEMORY_DYNAMIC_COMPONENTS`, permet de visualiser les différentes valeurs de chaque pool, entre autre la `shared_pool`, le `database buffer cache`, le `large pool`, etc ...

```
|Select component, current_size, min_size, max_size  
|from v$memory_dynamic_components ;
```

22.3.3 Nouveau cache en version 11g : result_cache

Ce cache est un nouveau composant de la SGA et est utilisé par Oracle pour initialiser le paramètre `MEMORY_TARGET`.

Par défaut ce paramètre est positionné à une valeur égale à 128K.

22.4 L'optimiseur Oracle

L'optimiseur de requêtes SQL permet de générer des plans d'exécution des requêtes performants. Par exemple utiliser un index si une table a de nombreuses lignes sera plus rapide que d'effectuer un balayage complet de table (FULL TABLE SCAN).

Pour que l'optimiseur de requêtes génère des plans d'exécutions optimales, des statistiques doivent être générées sur les objets (tables et index).

Les statistiques donnent à l'optimiseur des indications sur la volumétrie des objets mais ces statistiques sont également utilisées pour connaître la dégradation des objets (fragmentation, profondeur des index)



Autrefois (avant la version 9i), il fallait conserver une trace des objets pour déterminer si une collecte de statistiques était nécessaire. Si un objet n'avait pas de statistiques ou si elles étaient périmées, des plans d'exécution SQL erronés étaient générés et provoquait une dégradation des performances lors de l'exécution des requêtes.

Avec oracle 9i, si la supervision était utilisée, la commande `DBMS_STATS`, pouvait être utilisée pour collecter des statistiques.

```
DBMS_STATS.GATHER_SCHEMA_STATS(schema name, option=> 'GATHER AUTO')  
;
```

Cette commande générait des statistiques optimisées, en incluant des histogrammes, sur les objets pour lesquels, les statistiques étaient considérées comme périmées. Mais dans ce cas il fallait activer le monitoring et `DBMS_STATS` régulièrement.

Avec la version 10g, l'outil de statistiques automatise ces tâches et vous annonce s'il est nécessaire de générer des statistiques ou non.

Cette caractéristique réduit la probabilité d'exécuter du code SQL non performant causé par des statistiques inexistantes ou périmées.

22.4.1 Les optimiseurs RBO et CBO

L'optimiseur de requêtes RBO disparaît en version 9i. Cet optimiseur était basé sur des règles. Par exemple règle 1 si un index existe il faut l'utiliser.

RBO existe toujours dans la version 10g mais il n'est plus supporté, c'est-à-dire qu'aucun changement de code n'a été fait pour RBO et il n'y aura plus aucune correction des bugs.

Rappel de la version 9i

Le paramètre `OPTIMIZER_MODE` configure l'instance pour l'optimisation syntaxique RBO ou statistique CBO.

- ♦ `OPTIMIZER_MODE = CHOOSE` (CBO, valeur par défaut)
- ♦ `OPTIMIZER_MODE = RULE` (RBO)
- ♦ `OPTIMIZER_MODE = FIRST_ROWS` (CBO, avec l'objectif de minimiser le temps de réponse pour extraire la première ligne)
- ♦ `OPTIMIZER_MODE = ALL_ROWS` (CBO, avec l'objectif de minimiser le temps de réponse pour extraire toutes les lignes)

A partir de la version 10g seul l'optimiseur CBO est utilisé par Oracle. C'est un optimiseur basé sur les coûts. Cet optimiseur tient compte des statistiques générées dans la base pour générer les plans d'exécution des requêtes.

⇒ CBO est plus performant

A partir de la version 10g comme Oracle ne supporte que CBO, toutes les applications qui tournent sur la 10g doivent utiliser cet optimisateur (voir la notice oracle Metalink (189702.1) pour la décharge du support technique concernant le RBO). Cette notice fournit des détails sur l'abandon du support RBO et la migration des applications basées sur le RBO vers le CBO.



Voici certaines conséquences qu'il est possible de rencontrer :

- ♦ Les valeurs `CHOOSE` et `RULE` ne sont plus reconnues comme valeurs pour le paramètre d'initialisation `OPTIMIZER_MODE`. Les fonctionnalités de ces valeurs existent toujours mais seront supprimées dans une version future. Ceci est aussi valable pour les `HINTS` correspondants au `RBO`.
- ♦ `ALL_ROWS` est la valeur par défaut pour le paramètre d'initialisation `OPTIMIZER_MODE`.
- ♦ Les applications qui utilisent `RBO` doivent migrer vers `CBO`.

Dans la 1^{ère} release de Oracle 9i, le model de coût était limité seulement par les facteurs I/O. la 9i introduisait *CPU costing* pour rendre possible la comptabilisation des opérations CPU-intensive et CPU-only (CPU seule).

Par exemple une des opérations importante de CPU-only est d'extraire des données du buffer cache.

Aujourd'hui CBO a évolué et est devenu beaucoup plus performant.

Le « *SQL Tuning Advisor* » est une nouvelle fonctionnalité de l'optimiseur de requêtes depuis la version 10g, qui automatise l'ensemble des processus de Tuning du SQL.

En utilisant le nouvel optimiseur de requêtes CBO pour faire du Tuning le processus automatique remplace le Tuning manuel qui est complexe répétitif et consommateur de temps

22.4.2 Présentation du SQL Tuning Advisor

2 modes de fonctionnement :

- ⇒ Mode normal → l'utilisateur compile le SQL et génère le plan d'exécution.
 - le mode normale génère les plans d'exécution qui correspondent à la majorité des cas. Sous le mode normal l'optimiseur opère avec des contraintes de temps très strictes, une fraction de secondes pour trouver un bon plan d'exécution
- ⇒ Mode Tuning → l'optimiseur exécute une analyse supplémentaire pour vérifier si le plan d'exécution produit en mode normal peut être encore amélioré.
 - Le résultat de l'optimiseur de requête en mode Tuning n'est pas un plan d'exécution mais une série d'actions et leurs proratas de bénéfice attendus pour produire un plan bien meilleur. Quand il fonctionne dans le mode Tuning l'optimiseur est appelé ATO (*Automatic Tuning Optimizer*). Le *tuning* fait par ATO est appelé « *Tuning SQL Automatique* ».

Le « *Tuning automatique du SQL* » est une nouvelle fonctionnalité de l'optimiseur de requêtes qui automatise l'ensemble des processus de Tuning du SQL.

En utilisant le nouvel optimiseur de requêtes pour faire du Tuning le processus automatique remplace le Tuning manuel qui est complexe, répétitif et consommateur de temps.

Les caractéristiques du « *Tuning automatique du SQL* » sont disponibles via le *SQL Tuning Advisor*.

Dans le mode Tuning l'optimiseur peut prendre plusieurs minutes pour faire le Tuning d'une seule commande. L'ATO est destiné à être utilisé pour des requêtes SQL complexes avec un haut niveau de chargements qui ont un impact non trivial sur l'ensemble de la base.

Le *SQL Tuning Advisor* est en réalité le conducteur de processus de *tuning*. Il appelle l'ATO (*Automatic Tuning Optimizer*) pour exécuter les 4 types d'analyses suivantes :

- ♦ Statistics analysis (analyse des statistiques) : ATO vérifie chaque objet d'une requête afin d'identifier des statistiques manquantes ou figées (non générées depuis longtemps) et fournit des recommandations pour effectuer des statistique significatives. Il collecte aussi des informations



supplémentaires pour fournir les statistiques manquantes ou bien pour corriger les statistiques figées si les recommandations ne sont pas implémentées.

- ◆ SQL Profiling (créer un profile) : ATO vérifie ses propres estimations et collecte des informations supplémentaires pour corriger des erreurs d'estimation. Il collecte des informations sous forme de paramètres personnalisés de l'optimiseur comme « *First rows* » et « *All rows* » en s'appuyant sur l'historique d'exécution de la requête SQL. Il crée un profil SQL en utilisant ces informations et fait des recommandations pour la création de ce profil. Quand un profil SQL est créé, il permet à l'optimiseur de requêtes en mode normal de générer un plan ajusté.
- ◆ Acces path analysis (chemin d'accès) : ATO explore si un nouvel index peut être utilisé pour améliorer d'une manière significative l'accès à chaque table dans une requête et si c'est le cas il fait des recommandations pour la création de cet index.
- ◆ SQL structure Analysis : ATO tente d'identifier les commandes SQL qui utilisent de mauvais plans d'exécution et fait des recommandations pour les restructurer. La restructuration suggérée peut être au niveau syntaxique ou sémantique (la table existe, comment y accéder).

Le *tuning* SQL n'est pas simplement un des aspects les plus critiques pour la gestion des performances d'une base oracle mais est aussi une des tâches les plus difficiles à accomplir.

Avec oracle 10g la tâche d'identification de ces requêtes a été automatisée via l'*Automatic Database Diagnostic Monitor* (ADDM).

L'activité de *tuning* SQL représente une tâche continue car la charge SQL peut changer en fonction des nouveaux modules applicatifs déployés.

Le « *SQL Tuning Advisor* » est un nouvel outil destiné à remplacer le *tuning* manuel des requêtes SQL.

Les requêtes SQL qui consomment beaucoup de ressource (CPU, I/O et espace temporaire de travail) sont de bons candidats pour *SQL Tuning Advisor*.

L'outil reçoit une ou plusieurs requêtes SQL en entrée et fournit ensuite les éléments suivants :

- ⇒ des conseils sur l'optimisation des plans d'exécution
- ⇒ les gains estimés de performances
- ⇒ la commande effective pour implémenter ces conseils
- ⇒ les résultats de ce conseil

Vous pouvez choisir d'accepter ces conseils et faire le *tuning* du SQL via l'optimiseur oracle.

22.4.3 Impacte sur les Statistiques

L'optimisateur de requêtes se base sur les statistiques d'un objet pour générer des plans d'exécution.

Si ces statistiques n'existent pas ou sont figées, l'optimiseur n'a pas l'information nécessaire et peut générer des plans d'exécution aberrants.

L'ATO vérifie chaque objet utilisé par une requête afin de déterminer s'il y a des statistiques manquantes ou figées et produit 2 types de résultats :

- ◆ Des informations supplémentaires sous forme de statistiques pour les objets qui n'en ont pas et un facteur d'ajustement des statistiques pour les objets avec des statistiques obsolètes.
- ◆ Des recommandations pour produire des statistiques cohérentes pour des objets sans statistiques ou possédant des statistiques aberrantes.



22.5 L'optimiseur Oracle et la gestion des statistiques

La version Oracle 10g vous informe automatiquement des performances et ressources attribuées aux problèmes de performance.

Les statistiques sont automatiquement générées par le job `GATHER_STATS_JOB`. Ce job génère des statistiques sur tous les objets de la base de données.

Ce job est créé automatiquement à la création de la base et est managé par le *scheduler*. Il se déclenche tous les soirs à 22H00.

- Liste des jobs automatisés.

```
select owner, job_name, program_name,  
       schedule_name, enabled  
from dba_scheduler_jobs  
;
```

OWNER	JOB_NAME	PROGRAM_NAME	SCHEDULE_NAME	ENABL
SYS	PURGE_LOG	PURGE_LOG_PROG	DAILY_PURGE_SCHEDULE	TRUE
SYS	GATHER_STATS_JOB	GATHER_STATS_PROG	MAINTENANCE_WINDOW_GROUP	TRUE

`GATHER_DATABASE_STATS_JOB_PROC` attribue des priorités aux objets de la base de données afin que ceux qui en ont le plus besoin soient traités en priorité.

Si vous devez effectuer des chargements ponctuels de tables, régénérez manuellement les statistiques sur ces tables après chaque chargement.

L'optimiseur basé sur les coûts (CBO) est le seul optimiseur actif dans la version 10G.



CBO travaille en utilisant les statistiques générées sur les tables et les index. Alors il est fortement recommandé de ne pas laisser les objets qui sont souvent modifiés sans statistiques.

Ci-dessous sont présentés quelques conseils pour la collecte des statistiques sur le dictionnaire de données. Cette collecte se fait en utilisant le package `DBMS_STATS` :

- ⇒ **Régulièrement** : la méthode recommandée est d'utiliser soit `GATHER_DATABASE_STATS` ou `GATHER_SCHEM_STATS`, avec le paramètre `OPTION = GATHER AUTO`. Naturellement, ceci présume que la supervision est activée. Avec ce moyen seulement, les objets à réanalyser sont traités à chaque fois. La nouvelle procédure `GATHER_DICTIONARY_STATS` vous permet d'analyser le dictionnaire de données après un nombre suffisant d'opérations DDL.
- ⇒ **Cas particuliers** : n'oubliez pas d'analyser des objets indépendamment après une période de chargement caractéristique.

22.5.1 Statistiques sur les tables

Depuis la version 10g un job tourne en automatiquement de 22h à 6h les jours de la semaines et tout le week end, pour mettre à jour les statistiques des tables et des index de la base de données. C'est le job : `GATHER_STATS_JOB`.



Le job GATHER_STATS_JOB lance une procédure interne :

- ⇒ DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC (similaire à la procédure DBMS_STATS.GATHER_DATABASE_STATS avec option GATHER_AUTO) mais en priorisant les objets de la base de données, c'est-à-dire les objets ayant le plus besoin de statistiques.

Les **statistiques mettent à jour** les colonnes de la vue du dictionnaire de données DBA_TABLES et ses consœurs.

DBA_TABLES	
✓	NUM_ROWS = Nombre de lignes de la table.
✓	BLOCKS = Nombre de blocs sous la HWM (blocs utilisés).
✓	EMPTY_BLOCKS = Nombre de blocs au dessus de la HWM (blocs inutilisés).
✓	AVG_SPACE = Espace libre moyen en octets dans les blocs occupés (sous la HWM).
✓	AVG_ROW_LEN = Longueur moyenne d'une ligne en octets, en tenant compte des informations de contrôle (en-tête de lignes et en-tête de colonnes).
✓	CHAIN_CNT = Nombre de lignes chaînées ou migrées.
✓	SAMPLE_SIZE = Taille de l'échantillon utilisé en cas d'analyse réalisée sur la table.
✓	LAST_ANALYZED = Date et heure de la dernière analyse réalisée sur la table.

La gestion manuelle de la collecte des STATISTIQUES est faite en utilisant le package DBMS_STATS.

Si vous êtes connecté comme utilisateur SYS, vous pouvez collecter manuellement des statistiques sur tous les objets de la base avec la commande :

```
| EXEC DBMS_STATS.GATHER_DATABASE_STATS ;
```

Si vous êtes connecté comme utilisateur SYSTEM, vous pouvez collecter manuellement des statistiques sur les objets d'un schéma ou sur une table avec les commandes :

```
| DBMS_STATS.GATHER_TABLE_STATS('schema','table') ;
```

```
| DBMS_STATS.GATHER_SCHEMA_STATS('schema') ;
```

Exemples

```
| Exec dbms_stats.gather_schema_stats('charly');  
| Exec dbms_stats.gather_table_stats('charly','avion');
```



22.5.2 Interpréter les statistiques générées sur les tables

Les problèmes pouvant être détectés sont l'espace inutilisé alloué à une table et le faible taux d'occupation des blocs.

La génération de statistiques alimente les colonnes de la vue `DBA_TABLES`.

La valeur de `BLOCKS` est toujours exacte même si les statistiques sont inexactes ou manquantes.

Espace utilisé par une table

Espace inutilisé alloué à une table :

- ⇒ Le nombre de blocs inutilisés alloués à la table (`EMPTY_BLOCKS`) est important et la table ne va plus grossir (ou peu)
- ⇒ Lié à une clause `STORAGE` mal adaptée

Faible taux d'occupation des blocs :

Le rapport $(DB_BLOCK_SIZE - AVG_SPACE) / DB_BLOCK_SIZE$ est faible et les lignes actuelles ne vont pas grossir et peu de nouvelles lignes vont être insérées :

- ⇒ Lié à des valeurs `PCTFREE` et `PCTUSED` mal adaptées ou à une suppression importante de données, pose un problème de performance dans le parcours complet de la table

Le package DBMS_SPACE

Le package `DBMS_SPACE` possède plusieurs procédures qui permettent de superviser le stockage d'un segment.

Les principales procédures sont :

- ♦ `FREE_BLOCKS` : informations sur les blocs libres dans un segment dont l'espace est géré manuellement.
- ♦ `SPACE_USAGE` : informations sur l'occupation des blocs dans un segment dont l'espace est géré automatiquement.
- ♦ `UNUSED_SPACE` : informations sur les blocs inutilisés d'un segment.

Ce package possède d'autres procédures qui permettent d'estimer la taille d'un segment (table ou index), ou la tendance de croissance de celui-ci.

```
• liste des blocs vides et occupés dans la table AVION de CHARLY
Select t.blocks Occupes, s.blocks Alloues
From dba_tables t, dba_segments s
Where s.segment_name = t.table_name
And s.owner = t.owner
And t.table_name = 'AVION'
And t.owner = 'CHARLY'
;
```



22.5.3 Statistiques sur les index

Concernant les index et la dégradation de leur structure, le seul élément vraiment significatif est le nombre de *blocs de feuilles* qui doivent être lus.

Plus ce nombre est faible plus le nombre d'entrées-sorties le sera et plus grande sera la vitesse de lecture des lignes de tables.

⇒ Plus il y aura d'insertions ou de suppressions dans une table et plus le risque de fragmentation sera élevé.

Si les index sont analysés en même temps que les tables, le niveau de parallélisme qui peut s'appliquer au calcul des statistiques sur les tables ne s'applique pas aux index.

Si ceux-ci doivent être analysés en parallèle, il vaut mieux exécuter indépendamment la commande GATHER_INDEX_STATS.

```
Exec dbms.stats.gather_schema_stats (  
  'CHARLY',DBMS_STATS.AUTO_SAMPLE_SIZE  
)  
;  
Execute dbms_stats.gather_schema_statistics('clo01',  
20,estimate_percent=>20);
```

Dans l'analyse finale, vérifiez que les statistiques ont été calculées pour tous les index afin d'éviter des statistiques incomplètes dans la base de données.



Il faut savoir qu'un bloc d'index vide n'est pas réutilisé tant que l'index n'est pas compacté ou reconstruit.

Les **statistiques générées sur les index** alimentent les colonnes de la table DBA_INDEXES

DBA_INDEXES	
✓	BLEVEL, = Profondeur de l'arbre au niveau des branches (ne tient pas compte des feuilles). 0 si le bloc racine est égal au bloc feuille. Valeur exacte même en ESTIMATE
✓	LEAF_BLOCKS = Nombre de blocs feuilles dans l'index
✓	NUM_ROWS = Nombre de lignes dans l'index
✓	DISTINCT_KEY = Nombre de valeurs distinctes dans l'index
✓	SAMPLE_SIZE = Taille de l'échantillon utilisé en cas d'analyse ESTIMATE
✓	LAST_ANALYZED = Date et heure de la dernière analyse réalisée sur l'index



```
SQL> select INDEX_NAME, BLEVEL, LEAF_BLOCKS, NUM_ROWS
2  from dba_indexes
3  where index_name='PK_AVION'
4  ;
```

INDEX_NAME	BLEVEL	LEAF_BLOCKS	NUM_ROWS
PK_AVION	0	1	3



La hauteur d'un index est un élément clé de réduction du nombre d'entrées-sorties générées par cet index.

La génération de statistiques permet de donner suffisamment d'informations à l'optimiseur CBO sur les index.

22.5.4 Problèmes détectés sur les index

Deux problèmes peuvent être détectés :

- ⇒ Faible taux d'occupation et/ou profondeur importante de l'index

Profondeur importante de l'index

BLEVEL est élevé (supérieur à 5).

- ⇒ Lié à un PCTFREE mal adapté lors de la création ou à un index très volatile (beaucoup de mises à jour)
- ⇒ Dégrade les performances de l'utilisation de l'index.

22.6 Outil de collecte des statistiques

Pour que l'optimiseur de requêtes génère des plans d'exécution optimale, des statistiques doivent être générées sur les objets.

Autrefois (avant la version 9i), il fallait conserver une trace des objets pour déterminer si une collecte de statistiques était nécessaire. Si un objet n'avait pas de statistiques ou si elles étaient périmées, des plans d'exécution SQL erronés étaient générés.

Avec Oracle 9i, si la supervision était utilisée, la commande DBMS_STATS, pouvait être utilisée pour collecter des statistiques.

```
DBMS_STATS.GATHER_SCHEMA_STATS(schema name, option=> 'GATHER AUTO')
;
```

Cette commande génère des statistiques optimisées, en incluant des histogrammes, sur les objets pour lesquels, les statistiques étaient considérées comme périmées. Mais vous deviez activer le monitoring et DBMS_STATS régulièrement.



Avec la version 10g, l'outil de statistiques automatise ces tâches et vous annonce s'il est nécessaire de générer des statistiques.

Cette caractéristique réduit la probabilité d'exécuter du code SQL non performant causé par des statistiques inexistantes ou périmées.

22.6.1 GATHER_STATS_JOB

Par défaut `GATHER_STATS_JOB` est créé au moment de la création de la base de données, et exécute la procédure `DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC`. Il utilise le scheduler.

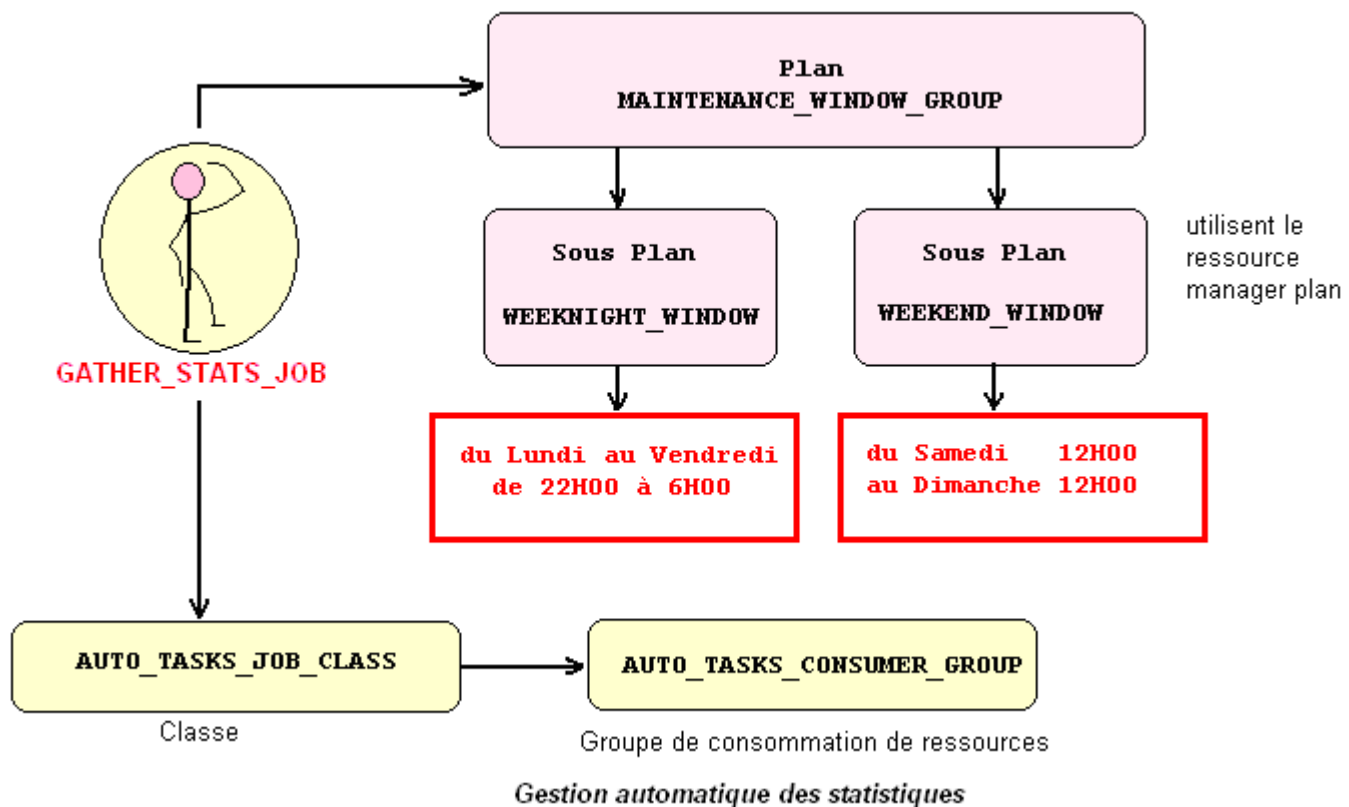
Deux fenêtres sont définies par défaut :

- ♦ `WEEKNIGHT_WINDOW` définie entre 22H00 et 6H00 tous les jours du lundi au vendredi
- ♦ `WEEKEND_WINDOW` définie du samedi midi au Dimanche midi

Un groupe de fenêtres appelées `MAINTENANCE_WINDOW_GROUP` est défini par défaut pour héberger ces fenêtres.

`GATHER_STATS_JOB` utilise une classe spécifique du scheduler appelée `AUTO_TASKS_JOB_CLASSE`. Cette classe est créée automatiquement et est associée à un groupe de consommateur de ressources appelé `AUTO_TASKS_CONSUMER_GROUP`.

Pour contrôler les ressources utilisées par `GATHER_STATS_JOB`, il suffit de définir un plan de gestion de ressources pour le `MAINTENANCE_WINDOW_GROUP` qui alloue les ressources pour l'`AUTO_TASKS_CONSUMER_GROUP`.



Pour que `GATHER_STATS_JOB` fonctionne correctement, vous devrez vous assurer que la valeur du paramètre d'initialisation : `STATISTIC_LEVEL` est égal à `TYPICAL`.



Si `GATHER_STATS_JOB` dépasse la fenêtre de temps allouée définie par `MAINTENANCE_WINDOW_GROUP`, le job continue jusqu'à ce qu'il soit terminé.

Vous devez générer des statistiques manuellement dans les cas suivants :

- ◆ Après une opération de chargement
- ◆ Pour les tables externes
- ◆ Pour collecter des statistiques systèmes
- ◆ Pour les objets en mémoire (*fixed objects*)

22.6.2 Modifier l'exécution des statistiques

Il est possible d'ajuster le temps d'ouverture des fenêtres de gestion prédéfinies. Par exemple, il est possible de changer l'intervalle de temps ou la fréquence de génération des statistiques. Il est également possible d'ajouter des plans de ressources à ces fenêtres pour contrôler les ressources utilisées par `GATHER_STATS_JOB`.

La page « Scheduler Windows » s'affiche. Dans cette page, apparaît une fenêtre :

⇒ cliquer sur le bouton « Edit » pour changer les paramètres.

A partir de cette page vous pouvez aussi ouvrir et fermer une fenêtre sélectionnée. Pour ce faire, sélectionnez l'action correspondante dans la liste déroulante de la fenêtre choisie puis :

⇒ cliquez sur « Go ».

Le bouton Modifier permet de modifier les paramètres.

Il est possible de désactiver la collecte automatique des statistiques en allant sur la page « Jobs » de l'onglet « Administration » puis désactiver `GATHER_STATS_JOB`.

22.7 Automatic Database Diagnostic Monitor (ADDM)

L'*Automatic Database Diagnostic Monitor* (ADDM), est un moteur d'autodiagnostic intégré directement dans la base de données Oracle.

L'ADDM met en œuvre les actions suivantes :

- ⇒ Il regarde l'ensemble du système
- ⇒ Il pose un diagnostic
- ⇒ puis propose des solutions ou vous envoie vers d'autres composants (par exemple *le SQL Tuning Advisor*)



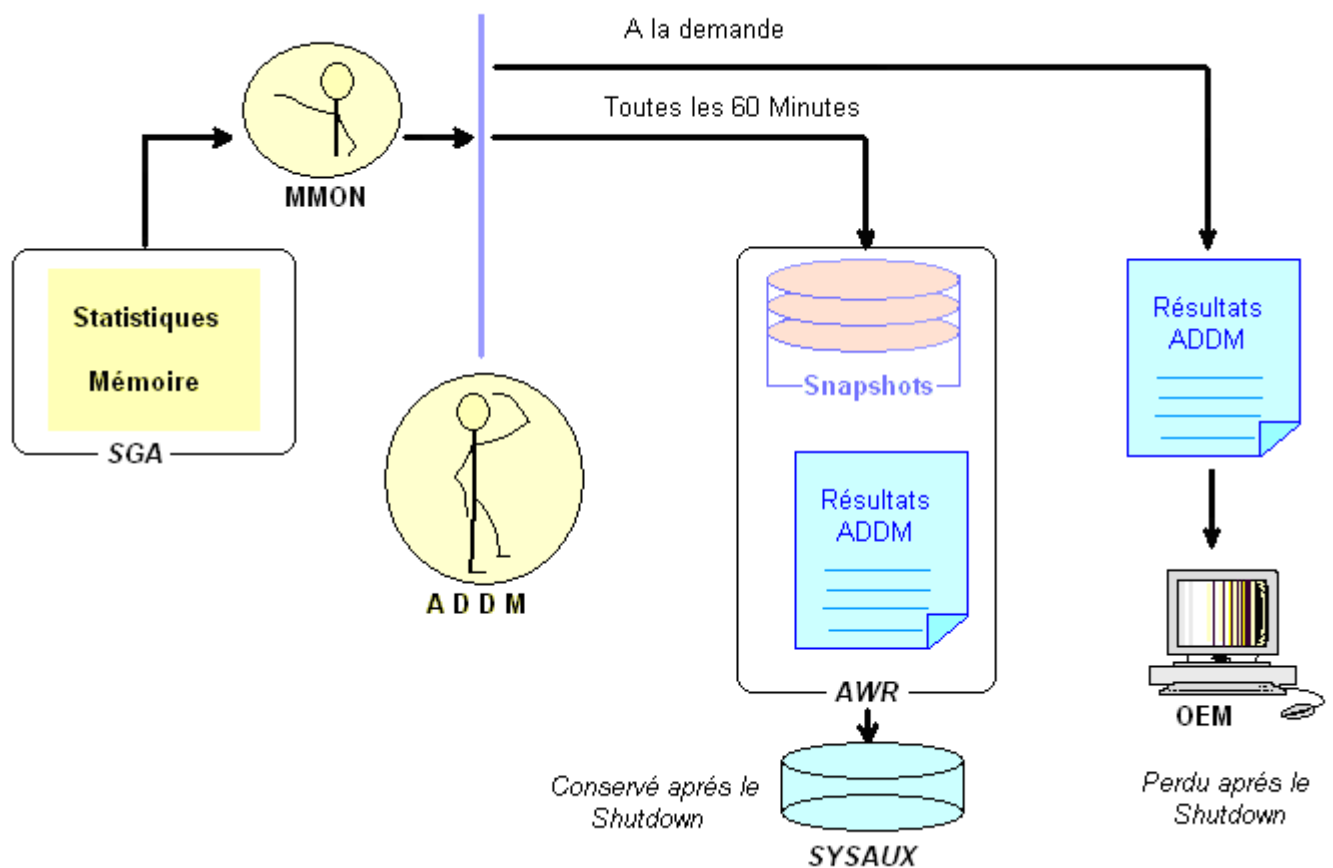
ADDM est appelé automatiquement par la base Oracle et effectue une analyse pour déterminer les problèmes potentiels principaux du système d'une manière préventive.

ADDM signale l'impact qu'un problème particulier peut avoir sur le système global. Si une recommandation est faite, ADDM précise l'amélioration attendue.

ADDM documente également les portions du système qui n'ont aucun problème.

La méthodologie utilisée par ADDM peut être appliquée sur tous les types de bases de données :

- ♦ OLTP
- ♦ Data Warehouse
- ♦ Environnements mixtes



Par défaut la base Oracle capte automatiquement des informations statistiques à partir de la SGA toutes les 60 minutes et les stocke dans l'Automatic Workload Repository (AWR) sous forme de *snapshots*.

Ces *snapshots* sont stockés sur le disque.

ADDM est programmé par le processus MMON pour tourner automatiquement sur chaque instance de la base afin de détecter les problèmes d'une manière préventive.

Chaque fois qu'un *snapshot* est créé, ADDM est activé pour faire une analyse de la période correspondant aux 2 derniers *snapshots*. Cette approche supervise d'une manière préventive l'instance et détecte les goulots d'étranglement avant qu'ils ne deviennent des problèmes conséquents.



Les résultats de chaque analyse ADDM sont stockés dans 1 'AWR (dans le tablespace SYSAUX) et sont aussi accessibles *via* le *Database Control*.

Il est possible d'invoquer manuellement une analyse d'ADDM sur la période définie entre 2 *snapshots* quelconques même si ADDM analyse la performance de la base Oracle sur la période définie par les 2 derniers *snapshots*.

22.7.1 Méthode d'analyse utilisée par ADDM

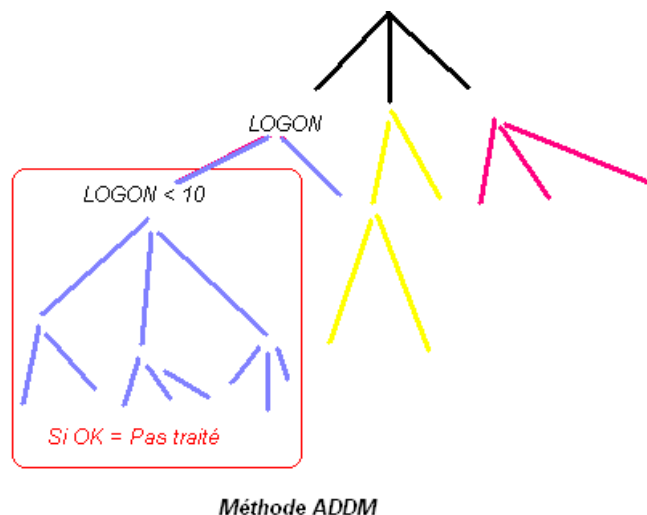
L'analyse ADDM utilise une approche *Top-Down* (de haut en bas) qui se focalise sur l'identification des goulots d'étranglements pour l'accès aux ressources.

Ceci est fait principalement en utilisant le nouveau modèle de statistiques de temps qui aide à déterminer la quantité de temps utilisée dans la base Oracle.

Cette méthode permet à ADDM d'identifier en premier le problème qui a l'impact le plus important sur toute la base.

En interne, ADDM utilise une structure arborescente pour représenter tous les problèmes possibles de *tuning*. L'arbre est basé sur le nouveau modèle de statistiques d'attente (*Wait*) utilisé par la base Oracle.

Cet arbre de classification est basé sur des décades d'expertises d'Oracle en tuning :



Le nœud initial de cet arbre, représente les symptômes ; en descendant vers les feuilles, ADDM identifie les problèmes principaux de performance.

ADDM parcourt l'arbre en utilisant des seuils de temps pour chaque nœud. Si le seuil de temps n'est pas dépassé pour un nœud en particulier, ADDM coupe le sous-arbre correspondant, alors il ne balaie pas l'arborescence sous ce nœud car si le nœud père ne pose pas de problème, alors les nœuds fils non plus. Cette structure arborescente permet à ADDM de déterminer efficacement la zone de recherche pour une identification rapide des problèmes.



L'exécution de l'analyse ADDM a un impact mineur sur le système et ne prend pas plus de 3 secondes pour s'exécuter.



Exemple

Examiner l'ouverture de sessions (LOGON) d'un système.
Si une des règles est que le ratio d'entrée en session ne doit pas dépasser 10 par seconde, alors en utilisant les données du nouveau modèle temps ADDM peut déterminer d'une manière quantitative que les connexions (LOGON) prennent 20% du temps de l'utilisation de la base oracle.
Ces problèmes étant quantifiés par ADDM (20% du temps) il ne vous reste plus qu'à les résoudre, puisque vous connaissez les valeurs quantifiées et les impacts sur le système.

22.7.2 Résultats de l'analyse ADDM dans le grid Control

Sur la page « Automatic Database Diagnostic Monitor (ADDM) » sont affichés les résultats détaillés de la dernière exécution d'ADDM.

Le « Database Time » représente la somme de temps actif du système par session dans la base de données pour la période d'analyse.

Un pourcentage spécifique est donné par la colonne « Impact » pour chaque diagnostic. Cet impact représente le temps consommé par le problème comparé au temps total utilisé par la base de données pour la période analysée.

Une liste d'icône représente le *snapshot* pour lequel un problème a été identifié. Il est possible de détailler d'autres diagnostics correspondants à d'autres *snapshots*.

Une liste des diagnostics est présentée en bas de page et donne un court résumé de ce qu'ADDM a trouvé comme points d'améliorations de performance pour l'instance analysée.

Oracle Enterprise Manager (SYSTEM) - Centre de conseil - Microsoft Internet Explorer

Adresse: http://poste02:1158/em/console/database/instance/advisorTasks?event=doload&dbPageNum=1&target=bora&type=oracle_database

Instance de base de données : bora > Centre de conseil

Page régénérée 8 févr. 2006 11 h 54 CET

Fonctions de conseil

- [ADDM](#)
- [Fonction de conseil sur la durée moyenne de récupération](#)
- [Fonction de conseil sur la mémoire](#)
- [Fonction de conseil sur les segments](#)
- [Gestion de l'annulation \(undo\)](#)
- [SQL Access Advisor](#)
- [SQL Tuning Advisor](#)

Tâches de la fonction de conseil

Rechercher

Sélectionnez un type de conseil et entrez éventuellement un nom de tâche pour filtrer les données affichées dans l'ensemble de résultats (ResultSet).

Type de conseil: Tous les types | Nom de tâche: | Exécutions de la fonction de conseil: Dernière exécution | Statut: Tout | Exécuter

Par défaut, la recherche renvoie toutes les correspondances en majuscules commençant par la chaîne saisie. Pour lancer une recherche exacte ou avec distinction maj/min, mettez la chaîne recherchée entre guillemets. Vous pouvez utiliser le caractère générique (%) dans une chaîne entre guillemets.

Résultats

Sélectionner	Type de conseil	Nom	Description	Utilisateur	Statut	Heure de début	Durée (secondes)	Expire dans (jours)
<input checked="" type="radio"/>	ADDM	ADDM:135680012_1_45	ADDM auto run: snapshots [44, 45], instance 1, database id 135680012	SYS	TERMINE	8 févr. 2006 11:00:34	0	30
<input type="radio"/>	Segment Advisor	SYS_AUTO_SPCADV_409822006	Auto Space Advisor	SYS	TERMINE	8 févr. 2006 10:00:04	1	30



22.7.3 Recommandations d'ADDM

Sur la page « Performance Finding Details », vous trouverez des recommandations pour résoudre le problème correspondant. Les recommandations sont regroupées par catégories comme : Schema, SQL Tuning, DB Configuration, et beaucoup d'autres. La colonne « Benefit (%) » affiche le temps d'exécution gagné par la base si cette recommandation est implémentée.

ADDM prend en compte un certain nombre de changements du système et ces recommandations peuvent inclure :

- ◆ Des changements Hardware : l'ajout de CPU ou le changement de la configuration du système d'entrée/sortie.
- ◆ La configuration de la base de données : changement des valeurs des paramètres d'initialisation.
- ◆ Un changement du schéma : partitionnement Hash d'une table ou d'un index ou l'utilisation de l'ASSM (*Automatic Segment Space Management*) qui correspond à la gestion automatique de la segmentation de l'espace.
- ◆ Des changements applicatifs : l'utilisation de l'option cache pour les séquences ou l'utilisation de variables BIND.

L'utilisation d'autres outils de conseil est possible : exécuter le *SQL Tuning Advisor* sur la base pendant une période de pointe d'exécution de requêtes SQL ou l'exécution du *Segment Advisor*.

22.7.4 Nouvelles vues en version 11g pour ADDM

En version 11g, ADDM voit apparaître de nouvelles vues du dictionnaire de données :

- DBA_ADDM_TASK contient la liste des tâches ADDM exécutées
- DBA_ADDM_INSTANCES informations sur les instances dans lesquelles ADDM a été exécuté
- DBA_ADDM_FINDING fournit des informations supplémentaires sur ADDM et son exécution
- DBA_ADDM_FDG_BREAKDOWN fournit des informations sur les performances de chaque instance
- DBA_ADDM_SYSTEM_DIRECTIVES informations concernant les directives (paramètres internes) prédéfinies
- DBA_ADDM_TASK_DIRECTIVES informations concernant les directives (paramètres internes) des tâches prédéfinies.

Le package DBMS_ADDM est optimisé en version 11g.

```
SQL> -- liste des tâches ADVISOR effectuées pour une journée donnée
SQL>
SQL> select task_id, to_char(created,'DD/MM/YYYY HH24:MI:SS') "Date Cree",
recommendation_count "NB recommendations"
2  from dba_advisor_tasks
3  where to_char(created,'DD/MM/YYYY HH24:MI:SS') > '30/07/2007 08:00:00'
4  order by "Date Cree";
```



TASK_ID	Date	Cree	NB recommendations
1428	30/07/2007	08:01:02	0
1429	30/07/2007	09:00:58	0
1430	30/07/2007	10:01:05	0
1431	30/07/2007	11:01:02	0
1432	30/07/2007	12:00:58	0
1433	30/07/2007	13:01:05	0
1434	30/07/2007	14:01:01	0
1435	30/07/2007	15:00:08	0
1436	30/07/2007	16:00:14	0
1437	30/07/2007	17:00:16	0
1438	30/07/2007	18:00:12	0
1439	30/07/2007	19:00:14	0
1440	30/07/2007	20:00:20	0
1441	30/07/2007	21:00:17	0
1442	30/07/2007	22:00:03	0
1443	30/07/2007	22:00:23	0
1444	30/07/2007	23:00:25	0
1445	31/07/2007	00:00:26	0
1446	31/07/2007	01:00:28	0
1447	31/07/2007	02:00:29	0
1448	31/07/2007	03:00:31	0
1449	31/07/2007	04:00:32	0
1450	31/07/2007	05:00:29	0

23 ligne(s) sélectionnée(s).

SQL>

SQL> spool OFF

22.7.5 Exemple de génération de rapport ADDM

```
SQL> @addmrpt
Current Instance
DB Id      DB Name      Inst Num Instance
-----
1885938199 LOWP01      1 LOWP01
Instances in this Workload Repository schema

DB Id      Inst Num DB Name      Instance      Host
-----
• 1885938199      1 LOWP01      LOWP01      aolfrdb-lm01
.webdb.aol.c
om

Using 1885938199 for database Id
Using      1 for instance number

Specify the number of days of snapshots to choose from
Entering the number of days (n) will result in the most recent
(n) days of snapshots being listed. Pressing <return> without
specifying a number lists all completed snapshots.

Listing the last 3 days of Completed Snapshots
Snap
Instance    DB Name      Snap Id      Snap Started    Level
-----
LOWP01      LOWP01      1350 29 Jul. 2007 00:0      1
1
1351 29 Jul. 2007 01:0      1
0
1352 29 Jul. 2007 02:0      1
0
-----liste des taches-----
Snap
Instance    DB Name      Snap Id      Snap Started    Level
-----
LOWP01      LOWP01      1378 30 Jul. 2007 04:0      1
0
```



```
1379 30 Juil. 2007 05:0    1
0
1380 30 Juil. 2007 06:0    1
0
1381 30 Juil. 2007 07:0    1
0
1382 30 Juil. 2007 08:0    1
0
1383 30 Juil. 2007 09:0    1
0
1384 30 Juil. 2007 10:0    1
0
1385 30 Juil. 2007 11:0    1
1
1386 30 Juil. 2007 12:0    1
0
1387 30 Juil. 2007 13:0    1
1
1388 30 Juil. 2007 14:0    1
1
1389 30 Juil. 2007 15:0    1
0
1390 30 Juil. 2007 16:0    1
0
Specify the Begin and End Snapshot Ids
Entrez une valeur pour begin_snap : 1382
Begin Snapshot Id specified: 1382
Entrez une valeur pour end_snap : 1384
End Snapshot Id specified: 1384

Specify the Report Name
The default report file name is addmrpt_1_1382_1384.txt. To use this name,
press <return> to continue, otherwise enter an alternative.
Entrez une valeur pour report_name : rpt_advisor.log
Using the report name rpt_advisor.log
Running the ADDM analysis on the specified pair of snapshots ...
Generating the ADDM report for this analysis ...
```

Exemple de rapport ADDM

```
DETAILED ADDM REPORT FOR TASK 'TASK_1640' WITH ID 1640

Analysis Period: 06-AOÛT -2007 from 11:00:19 to 15:00:25
Database ID/Instance: 1885938199/1
Database/Instance Names: LOWP01/LOWP01
Host Name: aolfrdb-lm01.webdb.aol.com
Database Version: 10.2.0.1.0
Snapshot Range: from 1553 to 1557
Database Time: 1304 seconds
Average Database Load: ,1 active sessions

FINDING 1: 44% impact (579 seconds)
SQL statements consuming significant database time were found.
RECOMMENDATION 1: SQL Tuning, 24% benefit (310 seconds)
ACTION: Investigate the SQL statement with SQL_ID "6nbtbvs44xdhj" for
possible performance improvements.
RELEVANT OBJECT: SQL statement with SQL_ID 6nbtbvs44xdhj
insert into EDITO_IMAGE (LAST_MODIFICATION_DATE, CREATED_BY,
CREATION_DATE, LAST_MODIFIED_BY, CONTENT, MIME_TYPE, AUTHOR,
CATEGORY, CHANNEL, COPYRIGHT, CONTENT DATE, END_DATE, DESCRIPTION,
KEYWORDS, PROMOTION, PURGE_DATE, REFERENCE_ID, SOURCE, START_DATE,
SUB_CATEGORY, ALT, GALLERY_ID, IMAGE_IN_GALLERY_INDEX, IMAGE_TEXT,
THUMBNAİL_ID, ID) values (:1, :2, :3, :4, :5, :6, :7, :8, :9, :10,
:11, :12, :13, :14, :15, :16, :17, :18, :19, :20, :21, :22, :23, :24,
:25, :26)
RATIONALE: SQL statement with SQL_ID "6nbtbvs44xdhj" was executed 94
times and had an average elapsed time of 3.2 seconds.
RATIONALE: Waiting for event "SQL*Net more data from client" in wait
class "Network" accounted for 93% of the database time spent in
processing the SQL statement with SQL_ID "6nbtbvs44xdhj".
```



```
----- suite du rapport -----
RECOMMENDATION 3: SQL Tuning, 6,1% benefit (79 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"aqzkzrwtzzfq8".
RELEVANT OBJECT: SQL statement with SQL_ID aqzkzrwtzzfq8 and
PLAN_HASH 3910361669
select this_.ID as ID7_0_, this_.CHILD_TYPE as CHILD2_7_0_,
this_.CHILD_ID as CHILD3_7_0_, this_.PARENT_TYPE as PARENT4_7_0_,
this_.PARENT_ID as PARENT5_7_0_, this_.RELATION as RELATION7_0_ from
CROSS_REFERENCE this_ where (this_.CHILD_TYPE=:1 and
this_.CHILD_ID=:2) and this_.RELATION=:3
RATIONALE: SQL statement with SQL_ID "aqzkzrwtzzfq8" was executed 75896
times and had an average elapsed time of 0.001 seconds.
RECOMMENDATION 4: SQL Tuning, 5,5% benefit (72 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"4gnvmykfv138v".
RELEVANT OBJECT: SQL statement with SQL_ID 4gnvmykfv138v and
PLAN_HASH 1107743244
select * from ( select this_.ID as ID0_1_,
this_.LAST_MODIFICATION_DATE as LAST2_0_1_, this_.CREATED_BY as
CREATED1_8_1_, this_.CREATION_DATE as CREATION2_8_1_,
this_.LAST_MODIFIED_BY as LAST3_8_1_, this_.CONTENT as CONTENT10_1_,
this_.MIME_TYPE as MIME2_10_1_, this_.AUTHOR as AUTHOR10_1_,
this_.CATEGORY as CATEGORY10_1_, this_.CHANNEL as CHANNEL10_1_,
this_.COPYRIGHT as COPYRIGHT10_1_, this_.CONTENT_DATE as
CONTENT7_10_1_, this_.END_DATE as END8_10_1_, this_.DESCRIPTION as
DESCRIPT9_10_1_, this_.KEYWORDS as KEYWORDS10_1_, this_.PROMOTION as
PROMOTION10_1_, this_.PURGE_DATE as PURGE12_10_1_, this_.REFERENCE_ID
as REFERENCE13_10_1_, this_.SOURCE as SOURCE10_1_, this_.START_DATE
as START15_10_1_, this_.SUB_CATEGORY as SUB16_10_1_,
this_.AUTHOR_EMAIL as AUTHOR1_25_1_, this_.JOKE_CAT_ID as
JOKE4_25_1_, this_.MODERATION as MODERATION25_1_, this_.TITLE as
TITLE25_1_, jokecatego2_.ID as ID0_0_,
jokecatego2_.LAST_MODIFICATION_DATE as LAST2_0_0_,
jokecatego2_.LOGICAL_ID as LOGICAL1_24_0_, jokecatego2_.NAME as
NAME24_0_ from DIV_JOKE this_ inner join DIV_JOKE_CATEGORY
jokecatego2_ on this_.JOKE_CAT_ID=jokecatego2_.ID where
this_.MODERATION=:1 order by this_.LAST_MODIFICATION_DATE desc )
where rownum <= :2
RATIONALE: SQL statement with SQL_ID "4gnvmykfv138v" was executed 7070
times and had an average elapsed time of 0.01 seconds.
RECOMMENDATION 5: SQL Tuning, 3,9% benefit (51 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"869fdws5s7k3w".
RELEVANT OBJECT: SQL statement with SQL_ID 869fdws5s7k3w and
PLAN_HASH 3366950133
select * from ( select showform0_.ID as col_0_0_ from DIV_SHOW
showform0_, RATING rating1_ where rating1_.TARGET_ID=showform0_.ID
group by showform0_.ID order by avg(rating1_.RATE) desc,
count(rating1_.ID) desc ) where rownum <= :1
RATIONALE: SQL statement with SQL_ID "869fdws5s7k3w" was executed 7938
times and had an average elapsed time of 0.0064 seconds.
----- suite du rapport -----
RECOMMENDATION 3: SQL Tuning, 5,5% benefit (72 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"4gnvmykfv138v".
RELEVANT OBJECT: SQL statement with SQL_ID 4gnvmykfv138v and
PLAN_HASH 1107743244
select * from ( select this_.ID as ID0_1_,
this_.LAST_MODIFICATION_DATE as LAST2_0_1_, this_.CREATED_BY as
CREATED1_8_1_, this_.CREATION_DATE as CREATION2_8_1_,
this_.LAST_MODIFIED_BY as LAST3_8_1_, this_.CONTENT as CONTENT10_1_,
this_.MIME_TYPE as MIME2_10_1_, this_.AUTHOR as AUTHOR10_1_,
this_.CATEGORY as CATEGORY10_1_, this_.CHANNEL as CHANNEL10_1_,
this_.COPYRIGHT as COPYRIGHT10_1_, this_.CONTENT_DATE as
CONTENT7_10_1_, this_.END_DATE as END8_10_1_, this_.DESCRIPTION as
DESCRIPT9_10_1_, this_.KEYWORDS as KEYWORDS10_1_, this_.PROMOTION as
PROMOTION10_1_, this_.PURGE_DATE as PURGE12_10_1_, this_.REFERENCE_ID
as REFERENCE13_10_1_, this_.SOURCE as SOURCE10_1_, this_.START_DATE
as START15_10_1_, this_.SUB_CATEGORY as SUB16_10_1_,
this_.AUTHOR_EMAIL as AUTHOR1_25_1_, this_.JOKE_CAT_ID as
JOKE4_25_1_, this_.MODERATION as MODERATION25_1_, this_.TITLE as
TITLE25_1_, jokecatego2_.ID as ID0_0_,
jokecatego2_.LAST_MODIFICATION_DATE as LAST2_0_0_,
```




```
jokecatego2.LOGICAL_ID as LOGICAL1_24_0_, jokecatego2.NAME as
NAME24_0_ from DIV JOKE this_ inner join DIV JOKE_CATEGORY
jokecatego2_ on this_.JOKE_CAT_ID=jokecatego2_.ID where
this_.MODERATION=:1 order by this_.LAST_MODIFICATION_DATE desc )
where rownum <= :2
RATIONALE: SQL statement with SQL ID "4gnvmykfv138v" was executed 7070
times and had an average elapsed time of 0.01 seconds.
RATIONALE: Average CPU used per execution was 0.01 seconds.
RECOMMENDATION 4: SQL Tuning, 3,9% benefit (51 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"869fdws5s7k3w".
RELEVANT OBJECT: SQL statement with SQL_ID 869fdws5s7k3w and
PLAN_HASH 3366950133
select * from ( select showform0_.ID as col_0_0_ from DIV SHOW
showform0_, RATING rating1_ where rating1_.TARGET_ID=showform0_.ID
group by showform0_.ID order by avg(rating1_.RATE) desc,
count(rating1_.ID) desc ) where rownum <= :1
RATIONALE: SQL statement with SQL ID "869fdws5s7k3w" was executed 7938
times and had an average elapsed time of 0.0064 seconds.
RATIONALE: Average CPU used per execution was 0.0064 seconds.
RECOMMENDATION 5: SQL Tuning, 3,5% benefit (46 seconds)
ACTION: Run SQL Tuning Advisor on the SQL statement with SQL_ID
"1w27xdpth04w3".
RELEVANT OBJECT: SQL statement with SQL_ID 1w27xdpth04w3 and
PLAN_HASH 979632492
select * from ( select this_.ID as ID0_0_,
this_.LAST_MODIFICATION_DATE as LAST2_0_0_, this_.CREATION_DATE as
CREATION1_1_0_, this_.HIDDEN as HIDDEN3_0_, this_.MODERATED as
MODERATED3_0_, this_.TEXT_CONTENT as TEXT3_3_0_, this_.USER_ID as
USER4_3_0_, this_.TARGET_TYPE as TARGET5_3_0_, this_.TARGET_ID as
TARGET6_3_0_ from REVIEW this_ where this_.TARGET_ID in (select
this0_.ID as y0_ from DIV MOVIE this0_) and this_.HIDDEN=:1 order
by this_.CREATION_DATE desc ) where rownum <= :2
RATIONALE: SQL statement with SQL ID "1w27xdpth04w3" was executed 7255
times and had an average elapsed time of 0.0063 seconds.
RATIONALE: Average CPU used per execution was 0.0063 seconds.
FINDING 4: 4,9% impact (64 seconds)
Wait event "SQL*Net more data to client" in wait class "Network" was consuming
significant database time.

RECOMMENDATION 1: Application Analysis, 4,9% benefit (64 seconds)
ACTION: Investigate the cause for high "SQL*Net more data to client"
waits. Refer to Oracle's "Database Reference" for the description of
this wait event.

RECOMMENDATION 2: Application Analysis, 4,9% benefit (64 seconds)
ACTION: Investigate the cause for high "SQL*Net more data to client"
waits in Service "SYS$USERS".
SYMPTOMS THAT LED TO THE FINDING:
SYMPTOM: Wait class "Network" was consuming significant database time.
(38% impact [491 seconds])
FINDING 5: 3,2% impact (42 seconds)
Soft parsing of SQL statements was consuming significant database time.
RECOMMENDATION 1: Application Analysis, 3,2% benefit (42 seconds)
ACTION: Investigate application logic to keep open the frequently used
cursors. Note that cursors are closed by both cursor close calls and
session disconnects.

RECOMMENDATION 2: DB Configuration, 3,2% benefit (42 seconds)
ACTION: Consider increasing the maximum number of open cursors a session
can have by increasing the value of parameter "open_cursors".
ACTION: Consider increasing the session cursor cache size by increasing
the value of parameter "session_cached_cursors".
RATIONALE: The value of parameter "open_cursors" was "700" during the
analysis period.
RATIONALE: The value of parameter "session_cached_cursors" was "20"
during the analysis period.

ADDITIONAL INFORMATION

Wait class "Application" was not consuming significant database time.
Wait class "Commit" was not consuming significant database time.
Wait class "Concurrency" was not consuming significant database time.
Wait class "Configuration" was not consuming significant database time.
```



Wait class "User I/O" was not consuming significant database time.
Session connect and disconnect calls were not consuming significant database time.
Hard parsing of SQL statements was not consuming significant database time.

The analysis of I/O performance is based on the default assumption that the average read time for one database block is 10000 micro-seconds.

An explanation of the terminology used in this report is available when you run the report with the 'ALL' level of detail.



23 La gestion des utilisateurs

Dans une base Oracle, les droits des utilisateurs sont gérés avec la notion de privilège.

Les privilèges peuvent être attribués directement aux utilisateurs ou par l'intermédiaire de rôles.

⇒ Un rôle est un regroupement nommé de privilèges

Un privilège est le droit

D'exécuter un ordre SQL (par exemple, créer une table) : ⇒ Privilège système
D'accéder à un objet
d'un autre utilisateur : ⇒ Privilège objet

Par défaut quand on crée un utilisateur Oracle ne lui alloue aucun privilège (*aucun droit*), pas même celui de se connecter. Ainsi la totalité des droits sont attribués explicitement en utilisant l'ordre GRANT.

La gestion des utilisateurs et de la sécurité permet :

- ⇒ De définir les utilisateurs qui peuvent se connecter à la base de données
 - Avec une identification par le système d'exploitation ou par la base de données
- ⇒ De définir dans quel tablespace par défaut, un utilisateur peut créer des objets
- ⇒ De limiter l'utilisation des ressources système
- ⇒ D'imposer une politique de gestion de mots de passe
 - Expiration périodique, non réutilisation avant un certain temps, ...
- ⇒ De définir les droits de chaque utilisateur à l'intérieur de la base de données
 - Droit de faire une action en général (par exemple, créer une table)
 - Droit de faire une action sur un objet spécifique (par exemple, mettre à jour les données d'une table)

23.1.1 Création d'un utilisateurs identifié par le système d'exploitation

L'utilisateur se connecte à la base sans saisir de nom ni de mot de passe

```
SQL> CONNECT /  
Connected.
```

Oracle ne vérifie pas le mot de passe mais contrôle simplement que le nom de l'utilisateur au niveau du système d'exploitation correspond à un nom d'utilisateur dans la base de données.

Pour faire le lien entre le nom de l'utilisateur dans le système d'exploitation et le nom de l'utilisateur dans la base de données, Oracle utilise un préfixe défini par le paramètre OS_AUTHENT_PREFIX (par défaut égal à OPS\$, mais peut être égal à une chaîne vide).

Exemple

L'utilisateur ayant pour nom « charly » au niveau du système d'exploitation ne pourra se connecter à la base par un CONNECT / que s'il existe un compte Oracle équivalent à ops\$charly.

- ⇒ Le préfixe peut être égal à une chaîne vide, dans ce cas le paramètre OS_AUTHENT_PREFIX = « ».
- ⇒ Le paramètre REMOTE_OS_AUTHENT peut en plus être positionné à TRUE pour indiquer si les utilisateurs distants peuvent être identifiés par cette méthode (FALSE pour l'interdire).

Sur plate forme Windows, le nom de domaine, ou le nom de la machine doivent faire partie du nom de l'utilisateur. De plus, vous devez rajouter une CLE dans votre base de registre.



- ⇒ Dans HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEid (ou id correspond à l'instance qui devra authentifier les utilisateurs à l'aide du système).
 - Rajouter la clé OSAUTH_PREFIX_DOMAIN (de type valeur de chaîne extensible) et lui donner la valeur TRUE s'il ne fait partie d'un domaine ou FALSE s'il fait parti d'un domaine.

Le paramètre REMOTE_OS_AUTHENT doit être positionné à TRUE pour indiquer si un utilisateur distant peut se connecter. Ce paramètre est déprécié en version 11.2.

Exemple

Configuration sur le serveur

La 1ère étape sera de créer un utilisateur OS local ou de domaine sur le serveur.

La 2ème étape sera de configurer la valeur du paramètre OS_AUTHENT_PREFIX (dans le fichier SPFILE).

La 3ème étape va dépendre du type d'utilisateur système que vous voulez autoriser.

Sur Windows, si l'utilisateur ne fais pas partie d'un domaine, vous devez rajouter une CLE dans votre base de registre.

- ⇒ Dans HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEid (ou id correspond à l'instance qui devra authentifier les utilisateurs à l'aide du système).
 - Rajouter la clé OSAUTH_PREFIX_DOMAIN (de type valeur de chaîne extensible) et lui donner la valeur TRUE.

La 4ème étape consistera à vérifier la valeur du paramètre SQLNET.AUTHENTICATION_SERVICES dans le fichier sqlnet.ora.

- ⇒ Celui-ci devra contenir la valeur NTS.

La 5ème étape consistera à créer explicitement le nouvel utilisateur de la base de données.

si l'utilisateur est un utilisateur OS local

```
-- On se connecte en tant que DBA
CONNECT system/<mot de passe>
-- Si l'utilisateur est un utilisateur local on lance cette commande
CREATE USER ops$charly IDENTIFIED EXTERNALLY;
-- On lui donne ensuite les droits par défaut
GRANT connect, resource TO ops$charly;
```

Si l'utilisateur est un utilisateur OS d'un domaine :

```
-- On se connecte en tant que DBA
CONNECT system/<mot de passe>
-- Si l'utilisateur est un utilisateur local on lance cette commande
CREATE USER "OPS$GALAXY\CHARLY" IDENTIFIED EXTERNALLY;
-- On lui donne ensuite les droits par défaut
GRANT connect, resource TO "OPS$GALAXY\CHARLY";
```



Attention

Dans le deuxième cas il faudra mettre le nom d'utilisateur en majuscule car des guillemets sont utilisés et Oracle devient alors sensible à la casse.

⇒ Si vous ne saisissez pas le login en majuscule vous aurez l'erreur ORA-01017.

Sous Unix, il sera très important de préfixer le \$ par un \ pour éviter que celui-ci ne soit interprété comme étant une variable d'environnement.

Le nouvel utilisateur pourra se connecter sur la base de données en utilisant la commande à partir du serveur.

```
| SQLPLUS /
```

Configuration sur le client

Les actions à effectuer pour que l'utilisateur puisse se connecter à distance sur le serveur tout en étant authentifié par son OS. (ce qui est le cas dans la majorité du temps)

La première chose à faire est de mettre à TRUE la valeur du paramètre REMOTE_OS_AUTHENT dans le fichier SPFILE. Si vous oubliez de mettre à TRUE ce paramètre vous obtiendrez alors les erreurs suivantes :

```
|ORA-01004: default username feature not supported; logon denied  
|ORA-01988: remote os logon is not allowed
```

Il faudra avoir un compte local ou un compte de domaine sur la machine cliente.

Vérifier la valeur du paramètre SQLNET.AUTHENTICATION_SERVICES dans le fichier sqlnet.ora.

⇒ Celui-ci devra contenir la valeur NTS.

Configurer le client Oracle Net pour pouvoir accéder à la base de données
Se connecter en utilisant la commande :

```
|SQLPLUS /@<chaîne de connection>
```

23.1.2 Création d'un utilisateurs identifié par Oracle

L'utilisateur se connecte à la base en saisissant un nom et un mot de passe

```
|SQL> CONNECT clo/defidba  
|Connected.
```

Oracle vérifie le nom et le mot de passe de l'utilisateur

L'ordre SQL CREATE USER permet de créer un nouvel utilisateur :

```
|CREATE USER nom IDENTIFIED { BY mot de passe | EXTERNALLY }  
|[ DEFAULT TABLESPACE nom tablespace ]  
|[ TEMPORARY TABLESPACE nom tablespace ]  
|[ QUOTA { valeur [K|M] | UNLIMITED } ON nom tablespace [,...] ]  
|[ PROFILE nom profil ]  
|[ PASSWORD EXPIRE ]  
|[ ACCOUNT { LOCK | UNLOCK } ] ;
```



- NOM = Nom de l'utilisateur, respecte les règles de nommage d'Oracle.
- IDENTIFIED = Indique si l'utilisateur est identifié par le système d'exploitation (EXTERNALLY) ou par Oracle (BY mot_de_passe).
- DEFAULT TABLESPACE = Indique dans quel tablespace les objets de l'utilisateur sont créés par défaut (c'est à dire si aucune clause TABLESPACE n'est présente lors de la création de l'objet dans la clause de stockage), si la clause est omise, le tablespace par défaut est le tablespace USERS.
- TEMPORARY TABLESPACE = Indique dans quel tablespace les segments temporaires de l'utilisateur (tris par exemple) sont créés. Si la clause est omise, le tablespace pour les segments temporaires est le tablespace TEMP.
- QUOTA = Indique dans quel(s) tablespace(s) l'utilisateur peut créer des objets, et jusqu'à quelle limite, Par défaut, l'utilisateur n'a aucun quota sur aucun tablespace.
- PROFILE = Profil attribué à l'utilisateur, si aucun profil n'est spécifié alors le profil DEFAULT est attribué par défaut. PASSWORD EXPIRE = Force une modification du mot de passe lors de la première connexion, (Sans objet si l'utilisateur est identifié par le système d'exploitation).
- ACCOUNT LOCK = le compte est verrouillé et la connexion interdite
- ACCOUNT UNLOCK = le compte n'est pas verrouillé et la connexion autorisée (par défaut)

- Utilisateur identifié par Oracle -

```
CREATE USER charly IDENTIFIED BY defidba  
PASSWORD EXPIRE ;
```



Pour qu'un nouvel utilisateur puisse effectivement se connecter, il faut en plus lui donner le droit de le faire, en lui attribuant le privilège système CREATE SESSION.

23.1.3 Modification d'un utilisateur dans Oracle

L'ordre SQL ALTER USER permet de modifier un utilisateur

Syntaxe

```
ALTER USER nom  
[ IDENTIFIED { BY mot de passe | EXTERNALLY } ]  
[ DEFAULT TABLESPACE nom tablespace ]  
[ TEMPORARY TABLESPACE nom tablespace ]  
[ QUOTA { valeur [K|M] | UNLIMITED } ON nom tablespace [,...] ]  
[ PROFILE nom profil ]  
[ PASSWORD EXPIRE ]  
[ ACCOUNT { LOCK | UNLOCK } ] ;
```

- Modification du mot de passe d'un utilisateur

```
ALTER USER charly  
IDENTIFIED BY defidba  
PASSWORD EXPIRE;
```



- Modification du tablespace par défaut et attribution de quotas

```
ALTER USER clo
  DEFAULT TABLESPACE test
  QUOTA UNLIMITED ON test
  QUOTA 10M ON data;
```
- Déverrouillage d'un compte

```
ALTER USER betty
  ACCOUNT UNLOCK;
```

23.1.4 Suppression d'un utilisateur

L'ordre SQL DROP USER permet de supprimer un utilisateur.

Si l'utilisateur possède des objets, l'option CASCADE doit être présente pour forcer la suppression préalable des objets. Si l'utilisateur possède des objets et que l'option CASCADE est absente, l'erreur ORA-01922 est retournée.

Un utilisateur connecté ne peut pas être supprimé.

```
| DROP USER nom [ CASCADE ] ;
```

Pour transférer des objets d'un utilisateur à un autre il faut exporter les objets qui lui appartiennent ; puis les réimportés dans un autre compte utilisateur. Vous pouvez ensuite supprimer l'utilisateur et ses objets :

```
| DROP USER charly CASCADE;
```

23.1.5 Supervision des utilisateurs

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les utilisateurs :

- DBA_USERS : informations sur les utilisateurs
- DBA_TS_QUOTAS : informations sur les quotas des utilisateurs

DBA_ USERS	
USERNAME	Nom de l'utilisateur
USER_ID	Identifiant de l'utilisateur
PASSWORD	Mot de passe (crypté)
ACCOUNT_STATUS	Status du compte (LOCKED, UNLOCKED, EXPIRED)
LOCK_DATE	Date de verrouillage
EXPIRY_DATE	Date d'expiration du mot de passe
DEFAULT_TABLESPACE	Tablespace de travail par défaut
TEMPORARY_TABLESPACE	Tablespace temporaire
CREATED	Date de création de l'utilisateur
PROFILE	profil



23.2 Les Profils

Un profil est un ensemble nommé de limitations de ressources qui peut être attribué à un utilisateur.

Les ressources suivantes peuvent être limitées :

- ⇒ Temps CPU par appel et/ou par session
- ⇒ Nombre de lectures logiques par appel et/ou par session
- ⇒ Nombre de sessions ouvertes simultanément par un utilisateur
- ⇒ Temps d'inactivité par session
- ⇒ Durée totale de la session
- ⇒ Quantité de mémoire privée dans la SGA (configuration multithreaded server uniquement)

Depuis la version 8, les profils peuvent aussi être utilisés pour mettre en œuvre une politique de gestion des mots de passe approfondie.

Les fonctionnalités suivantes peuvent être mises en œuvre :

- ⇒ Verrouillage de compte (et durée de verrouillage) au delà d'un certain nombre d'échecs de tentative de connexion.
- ⇒ Durée de vie des mots de passe (avec éventuellement une période de grâce). La demande de changement de mot de passe est activée en version 11.2 par défaut au bout de 6 mois.
- ⇒ Non réutilisation d'un mot de passe avant un certain temps ou avant un certain nombre de changements
- ⇒ Complexité du mot de passe

23.2.1 Activer la limitation des ressources

Par défaut, le contrôle de la limitation des ressources n'est pas activé, créer des profils et les affecter aux utilisateurs n'a aucun effet sur les ressources !

Pour activer le contrôle de la limitation des ressources, il faut passer le paramètre RESOURCE_LIMIT à TRUE (FALSE par défaut).

- ⇒ RESOURCE_LIMIT = TRUE

Si la base est déjà ouverte, par un ALTER SYSTEM

- ⇒ ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;

Par contre, les fonctionnalités de gestion des mots de passe fonctionnent même si le paramètre RESOURCE_LIMIT est à FALSE (!).



23.2.2 Création d'un profil

L'ordre SQL CREATE PROFILE permet de créer un profil dans Oracle.

```
CREATE PROFILE nom LIMIT
[ SESSIONS PER USER { valeur | UNLIMITED | DEFAULT } ]
[ CPU PER SESSION { valeur | UNLIMITED | DEFAULT } ]
[ CPU PER CALL { valeur | UNLIMITED | DEFAULT } ]
[ CONNECT TIME { valeur | UNLIMITED | DEFAULT } ]
[ IDLE TIME { valeur | UNLIMITED | DEFAULT } ]
[ LOGICAL READS PER SESSION { valeur | UNLIMITED | DEFAULT } ]
[ LOGICAL READS PER CALL { valeur | UNLIMITED | DEFAULT } ]
[ COMPOSITE LIMIT { valeur | UNLIMITED | DEFAULT } ]
[ PRIVATE SGA { valeur [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED LOGIN ATTEMPTS { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD LIFE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD REUSE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD REUSE MAX { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD LOCK TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD GRACE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nom_fonction | NULL | DEFAULT } ] ;
```

- Session_per_user : nombre de sessions simultanées (utilisé)
- Cpu_per_session : CPU totale par session (1/100 s)
- Cpu_per_call : CPU totale par appel (1/100 s)
- Connect_time : durée totale de connexion (minutes) (utilisé)
- Idle_time : durée d'inactivité (minutes) (utilisé)
- Logical_reads_per_session : nombre de lectures logiques par session
- Logical_reads_per_call : nombre de lectures logiques par appel
- Private_sga : quantité de mémoire privée dans la SGA
- Composite_limit : somme pondérée de CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION et PRIVATE SGA. (pas utilisé)
- Failed_login_attempts : nombre d'échecs de tentative de connexion autorisés avant verrouillage du compte
- Password_lock_time : durée du verrouillage (jours). Il est possible de spécifier des heures en faisant = 1/24 = 1 heure)
- Password_life_time : durée de vie du mot de passe (jours)
- Password_grace_time : période de grâce après expiration du mot de passe (jours)
- Password_reuse_time : nombre de jours pendant lesquels un mot de passe ne peut pas être réutilisé
- Password_reuse_max : nombre de changements de mots de passe avant qu'un mot de passe puisse être réutilisé
- Password_verify_function : fonction de vérification du mot de passe (écrite en PL/SQL, par le DBA), une valeur = NULL permet de ne pas spécifier de fonction

Spécification des valeurs UNLIMITED et DEFAULT :

- ⇒ UNLIMITED : aucune limitation
- ⇒ DEFAULT : le paramètre hérite de la valeur du profil nommé DEFAULT



Exemple

```
CREATE PROFILE exploit LIMIT
    SESSIONS_PER_USER 3
    IDLE_TIME 30
    FAILED_LOGIN_ATTEMPTS 3
    PASSWORD_LIFE_TIME 30
    PASSWORD_REUSE_TIME 180
    PASSWORD_LOCK_TIME UNLIMITED
    PASSWORD_GRACE_TIME 3
    PASSWORD_VERIFY_FUNCTION verif_mdp_exploitation ;
```

Le profil nommé DEFAULT existe dès la création de la base de données, avec des valeurs UNLIMITED.

⇒ En version 11.2 ce profil oblige la modification du mot de passe tous les 6 mois.

Le profil DEFAULT est attribué par défaut aux utilisateurs à qui aucun profil n'est attribué.

Le profil DEFAULT peut être modifié par l'administrateur.

⇒ A un impact sur les utilisateurs qui ont le profil DEFAULT a un impact sur les profils qui ont des paramètres à DEFAULT



Le script %oracle_home%\rdbms\admin\utlpwmg.sql, contient un exemple de fonction de vérification qui est affecté au profil par défaut.

23.2.3 Modification d'un profil

L'ordre SQL ALTER PROFILE permet de modifier un profil :

```
ALTER PROFILE nom LIMIT
[ SESSIONS PER USER { valeur | UNLIMITED | DEFAULT } ]
[ CPU PER SESSION { valeur | UNLIMITED | DEFAULT } ]
[ CPU PER CALL { valeur | UNLIMITED | DEFAULT } ]
[ CONNECT TIME { valeur | UNLIMITED | DEFAULT } ]
[ IDLE TIME { valeur | UNLIMITED | DEFAULT } ]
[ LOGICAL READS PER SESSION { valeur | UNLIMITED | DEFAULT } ]
[ LOGICAL READS PER CALL { valeur | UNLIMITED | DEFAULT } ]
[ COMPOSITE LIMIT { valeur | UNLIMITED | DEFAULT } ]
[ PRIVATE SGA { valeur [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED LOGIN ATTEMPTS { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD LIFE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD REUSE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD REUSE MAX { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD LOCK TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD GRACE TIME { valeur | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nom_fonction | NULL | DEFAULT } ] ;
```



```
Modification du profil DEFAULTALTER PROFILE default LIMIT
SESSIONS_PER_USER 3
IDLE_TIME 30
FAILED_LOGIN_ATTEMPTS 5;
-- les autres paramètres gardent la valeur par défaut (UNLIMITED)

Modification d'un autre profil ALTER PROFILE exploit LIMIT
SESSIONS_PER_USER 5 -- passe de 3 à 5
IDLE_TIME UNLIMITED--suppression de la limite
FAILED_LOGIN_ATTEMPTS DEFAULT; -- prend la valeur par défaut (5)
-- le reste est inchangé
```



La modification d'un profil n'affecte les utilisateurs qu'à leur prochaine connexion.

Un profil est affecté à un utilisateur, soit lors de la création de l'utilisateur (CREATE USER), soit lors d'une modification de l'utilisateur (ALTER USER).

L'affectation d'un nouveau profil à des utilisateurs ne prend effet qu'à leur prochaine connexion

```
Lors de la création
CREATE USER xgeo IDENTIFIED BY tempo
TEMPORARY TABLESPACE temp
PROFILE exploitation
PASSWORD EXPIRE;

Lors d'une modification
• affectation d'un profil
ALTER USER oheu PROFILE exploitation;
-- ré-affectation du profil DEFAULT
ALTER USER oheu PROFILE DEFAULT;
```

23.2.4 Suppression d'un profil

L'ordre SQL DROP PROFILE permet de supprimer un profil :

```
DROP PROFILE nom [ CASCADE ] ;
```

Si le profil est attribué à des utilisateurs, l'option CASCADE doit être présente.

```
DROP PROFILE exploitation CASCADE;
```

Le profil DEFAULT est affecté en remplacement aux utilisateurs concernés.

La suppression d'un profil n'affecte les utilisateurs qu'à leur prochaine connexion.

Le profil DEFAULT ne peut pas être supprimé.



23.2.5 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les profils :

- DBA_USERS : informations sur les utilisateurs (è profil attribué) DBA_PROFILES : informations sur les profils

DBA_ PROFILES	
PROFILE	Nom du profil
RESOURCE_NAME	Nom de la ressource contrôlée
RESOURCE_TYPE	Type de la ressource contrôlée (KERNEL ou PASSWORD)
LIMIT	Limite de la ressource

DBA_ USERS	
USERNAME	Nom de l'utilisateur
USER_ID	Identifiant de l'utilisateur
PASSWORD	Mot de passe (crypté)
ACCOUNT_STATUS	Statut du compte (LOCKED, UNLOCKED, EXPIRED)
LOCK_DATE	Date de verrouillage
EXPIRY_DATE	Date d'expiration du mot de passe
DEFAULT_TABLESPACE	Tablespace de travail par défaut
TEMPORARY_TABLESPACE	Tablespace temporaire
CREATED	Date de création de l'utilisateur
PROFILE	profile

Dans cette vue, un profil est présenté avec plusieurs lignes, il y a une ligne par ressource contrôlée.

23.3 Le gestionnaire de ressources

Si vous souhaitez contrôler la limitation des ressources de façon plus précise qu'en utilisant des profils, vous pouvez utiliser le package DBMS_RESSOURCE_MANAGER appelé *Database Ressource Manager*.

Le package DBMS_RESSOURCE_MANAGER, apparu en version 10g, introduit des nouvelles méthodes d'allocation pour les procédures CREATE_CONSUMER_GROUP et CREATE_PLAN.



Ce package permet de définir l'utilisation de CPU par un groupe d'utilisateurs.

- ⇒ Vous spécifiez une méthode d'allocation pour la distribution de la CPU parmi les sessions du Consumer Group avec l'option CPU_MTH de la procédure CREATE_CONSUMER_GROUP.
Des planifications de type ROUND-ROBIN : assurent que les sessions soient exécutées d'une façon équitable, par conséquent la valeur par défaut du CPU_MTH est « Round-robin ».
La méthode d'allocation de type « Round-to-completion » : spécifie que les sessions ayant un temps d'activité plus important, seront planifiées en priorité par rapport aux autres sessions.
- ⇒ La méthode d'allocation de ressource pour spécifier combien de CPU sera allouée à chaque Consumer Group ou « *Subplan* », est spécifié avec la variable CPU_MTH de la procédure CREATE_PLAN.
- ⇒ La méthode par défaut pour des plans à plusieurs niveaux qui utilisent des pourcentages pour définir combien de CPU est distribuée aux différents Consumer Group est la méthode « *Emphasis* ».

23.3.1 Les groupes de consommation de ressources : Consumers Groups

Comme il est possible pour une session d'être associée à différents *Consumer Groups* de la base de données, le *Ressource Manager* utilise les niveaux de priorité de chaque attribut des *Consumers Groups* afin de résoudre les éventuelles ambiguïtés.

Un exemple d'une telle association est de prendre une session de l'utilisateur « *pdml* » et le placer dans le consumer group *DSS_GROUP*.

```
DBMS_CONSUMER_GROUP.SET_CONSUMER_GROUP_MAPPING
(
  DBMS_RESOURCE_MANAGER.ORACLE_USER ,
  'CLO', 'DSS_GROUP'
) ;
DBMS_CONSUMER_GROUP.SET_CONSUMER_GROUP_MAPPING
(
  DBMS_RESOURCE_MANAGER.ORACLE_USER ,
  'CHARLY', 'GROUP2'
) ;
DBMS_CONSUMER_GROUP.SET_CONSUMER_GROUP_MAPPING
(
  DBMS_RESOURCE_MANAGER.CLIENT_OS_USER ,
  'BILL', 'MANAGER_GROUP'
) ;
```

Dans l'exemple précédent, si une session est ouverte les actions suivantes sont effectuées :

- ♦ Si le nom du user est « CLO », le Ressource Manager assigne la session au Consumer Group : *DSS_GROUP*
- ♦ Si le user est « CHARLY » alors le Ressource Manager assigne la session au Consumer Group : *GROUP2*
- ♦ Si le nom du user est un user-OS client est BILL le Ressource Manager assigne la session au Consumer Group : *Manager_Group*.
- ♦ Toutefois, si l'utilisateur est entré en session sur la machine cliente comme BILL (attribut *CLIENT_OS_USER*) et se connecte à la base comme « *tp1* » (attribut *ORACLE_USER*), une ambiguïté est créée.



Les priorités par défaut des attributs sont définies de telle manière que le nom du *USER Oracle* est prioritaire par rapport au nom du *USER client-OS*. Dans ce cas le *Ressource Manager* va assigner la session au *Consumer Group* : GROUP2.

23.3.2 Le package DBMS_RESSOURCE_MANAGER

Le temps d'activité correspond au temps durant lequel une session d'un *Consumer Group* est en train d'exécuter une action ou d'être en attente d'entrée/sortie. Ce temps calculé se situe au niveau de la session et est déterminé en interne.

```
-- crer un Consumer Group
--
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP
(
  CONSUMER_GROUP => 'DSS GROUP' ,
  CPU_MTH        => 'RUN TO COMPLETION' ,
  COMMENT        => 'Temps actif important'
) ;
-- créer un plan
--
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP
(
  PLAN => 'PLAN TEST' ,
  CPU_MTH => 'RATIO' ,
  COMMENT => 'Plan'
) ;
```

Les procédures CREATE_PLAN_DIRECTIVE dans l'exemple suivant créent un plan avec les ratios 10 ; 5 ; 3 et 2 pour les consumer group : GOLD_CG ; SILVER_CG ; BRONZE_CG et OTHER_GROUPS.

Ceci assure que :

- ♦ GOLD_CG obtient 10/20 des ressources CPU,
- ♦ SILVER_CG obtient 5/20,
- ♦ BRONZE_CG obtient 3/20,
- ♦ OTHER_GROUPS obtient 2/20.
- ♦ S'il y a des sessions actives seulement dans les Consumer Groups GOLD_CG et SILVER_CG alors GOLD_CG contiendra 10/15 des ressources CPU et SILVER_CG obtiendra 5/15.

23.3.3 Consumers groups et plan de ressources

```
DBMS_RESSOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(
  PLAN                => 'PLAN_TEST' ,
  GROUP_OR_SUBPLAN    => 'GOLD_CG' ,
  CPU_P1              => 10
) ;
DBMS_RESSOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(
  PLAN                => 'PLAN_TEST' ,
  GROUP_OR_SUBPLAN    => 'SILVER_CG' ,
  CPU_P1              => 5
) ;
```



```
DBMS_RESSOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(
  PLAN                => 'PLAN_TEST' ,
  GROUP_OR_SUBPLAN    => 'BRONZE_CG' ,
  CPU_P1              => 3
) ;
DBMS_RESSOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(
  PLAN                => 'PLAN_TEST' ,
  GROUP_OR_SUBPLAN    => 'OTHER_GROUPS' ,
  CPU_P1              => 2
) ;
```

Il y a plusieurs façons de gérer et surveiller le *Ressource Manager* en utilisant le OEM Database Control.

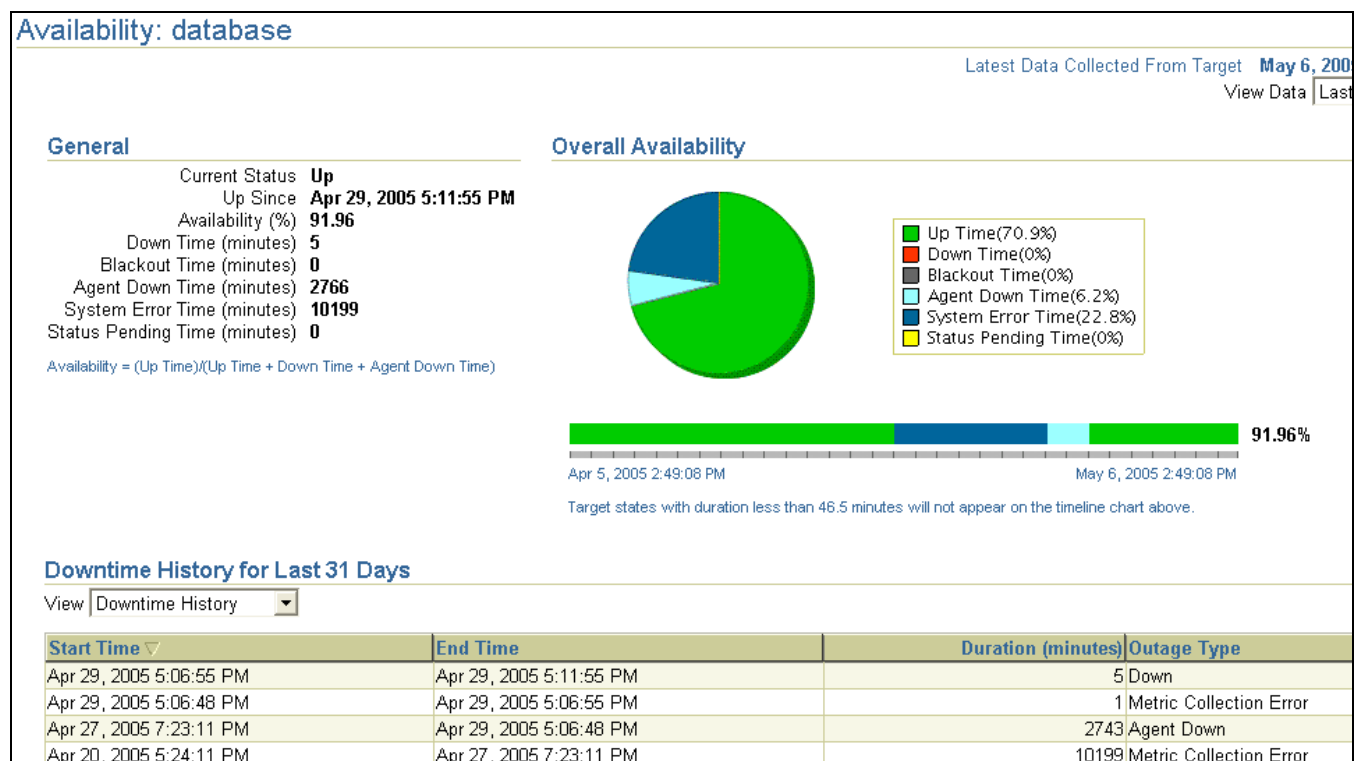
- ➡ A partir de l'onglet « Administration » cliquez sur le lien « Resource Monitors » dans la section « Ressource Manager ».

L'utilisation du *Diagnostic Pack* (ADDM) permet le suivi des performances et utilisation des ressources de la base de données.

Organisée en trois sections, la page des performances de la base de données affiche les informations de la machine, l'activité des utilisateurs et les informations de traitement sur un seul écran afin de faciliter les corrélations.

Le statut de la base de données peut être observé grâce aux graphiques des sessions actives qui montrent combien de CPU les utilisateurs consomment et si les utilisateurs attendent certaines ressources.

La page présente un graphique d'utilisation qui peut être utilisé pour déterminer si les traitements sont limités par certaines des ressources machines, par l'utilisation de la CPU ou par des contentions.



23.3.4 Assigner des priorités

Dans l'exemple suivant on modifie l'attribut ORACLE_USER à la valeur 9 quand la valeur par défaut était 7 et la valeur de l'attribut CLIENT_OS_USER à 7 alors qu'elle était à 9 par défaut.

Les valeurs donnés dans l'exemple sont les valeurs par défaut utilisées pour les associations des *Consumers Groups*.

```
DBMS_CONSUMER_GROUP.SET_CONSUMER_GROUP_MAPPING_PRI
(
  EXPLICIT           => 1 ,      -- plus grande priorité
  SERVICE_MODULE     => 2 ,
  SERVICE_MODULE_ACTION => 3 ,
  MODULE_NAME_ACTION => 4 ,
  MODULE_NAME        => 5 ,
  SERVICE_NAME       => 6 ,
  ORACLE_USER        => 9 ,
  CLIENT_PROGRAM      => 8 ,
  CLIENT_OS_USER      => 7 ,
  CLIENT_MACHINE      10      -- plus petite priorité
) ;
```

Les attributs MODULE_NAME et MODULE_NAME_ACTION sont très utiles pour les applications du middle-tiers (architecture 3-Tiers) qui utilisent le même nom de user pour tous les clients mais définissent leurs modules et leurs actions de façon identique dans des opérations différentes.

Exemple

Oracle Database 11g Release 2, permet de fixer le nombre maximum de processeurs utilisables par une instance en utilisant la fonctionnalité « *d'instance caging* », la seconde est liée à l'utilisation du paramètre MAX_UTILIZATION_LIMIT du *Ressource Manager* qui fixe une limite supérieure d'utilisation de la CPU pour un groupe d'utilisateur rattaché à un plan de ressources.

Pour cela, il faut :

- ⇒ Créer un plan de limitation de ressource avec une valeur maximale d'utilisation de la CPU.

Le script ci-dessous crée un plan de ressources qui limite à 17% la CPU dans le groupe PRIMO et à 24% la CPU dans le groupe SECONDO.

Ce script positionne également le plan d'exécution comme le plan actif et autorise l'utilisateur CHARLY à utiliser le groupe PRIMO :

```
begin
  dbms_resource_manager.clear_pending_area;
  dbms_resource_manager.create_pending_area;

  dbms_resource_manager.create_plan(
    plan      => 'LIMITED_CPU',
    comment   => 'Limited Amount of CPU');

  dbms_resource_manager.create_consumer_group(
    consumer_group => 'PRIMO',
    comment       => 'PRIMO Consumer Group',
    mgmt_mth      => 'ROUND-ROBIN');

  dbms_resource_manager.create_plan_directive(
    plan          => 'LIMITED_CPU',
    group_or_subplan => 'PRIMO',
```



```
comment          => 'PRIMO - Max to 25% of CPU',
max_utilization_limit => 17);

-- Create resource plan directives
dbms_resource_manager.create_plan_directive(
    plan          => 'LIMITED_CPU',
    group_or_subplan => 'SECONDO',
    comment       => 'SECONDO - Max to an additional 24% of CPU',
    max_utilization_limit => 24);

-- Submit the changes
dbms_resource_manager.validate_pending_area;
dbms_resource_manager.submit_pending_area;

end;
/

alter system set resource_manager_plan=LIMITED_CPU;

begin
    dbms_resource_manager_privs.grant_switch_consumer_group(
        grantee_name => 'CHARLY',
        consumer_group => 'PRIMO',
        grant_option  => false);
end;
/
```

Confirmer le paramétrage.

Pour confirmer le paramétrage du gestionnaire de ressources, vous pouvez exécuter les requêtes ci-dessous :

```
col plan format a12
col comments format a25
select plan, comments
  from dba_rsrc_plans
 where plan='LIMITED_CPU';

PLAN          COMMENTS
-----
LIMITED_CPU   Limited Amount of CPU

col group_or_subplan format a16
select plan, group_or_subplan, type, mgmt_p1, max_utilization_limit
  from dba_rsrc_plan_directives
 where plan='LIMITED_CPU';

PLAN          GROUP_OR_SUBPLAN TYPE          MGMT_P1 MAX_UTILIZATION_LIMIT
--
LIMITED_CPU   PRIMO             CONSUMER_GROUP          0          17
LIMITED_CPU   SECONDO            CONSUMER_GROUP          0          24
```




```
col consumer_group format a15
col mgmt_method format a15
col status format a6
col mandatory format a6
select consumer_group, mgmt_method, status, mandatory
  from dba_rsrc_consumer_groups
 where consumer_group in ('SECONDO','PRIMO');

CONSUMER_GROUP  MGMT_METHOD      STATUS MANDAT
-----
PRIMO           ROUND-ROBIN          NO
SECONDO        ROUND-ROBIN          YES
```

Observer les indicateurs clés

Pour savoir comment se comporte les utilisateurs actifs dans le groupe PRIMO, nous allons effectuer un test; pour cela notez l'identifiant de session grace à : sys_context :

```
sqlplus CHARLY/charly

select sys_context('userenv', 'sid') sid
  from dual;

SID
-----
125
```

Ouvrez une seconde session sous SYS et affectez la session de CHARLY dans le groupe de consommateur PRIMO.

```
sqlplus / as sysdba

set lines 80
col sid format 999999
col serial# format 999999
col username format a8
col resource_consumer_group format a25
select sid, serial#, username, resource_consumer_group
  from v$session where sid=125;

SID SERIAL# USERNAME RESOURCE_CONSUMER_GROUP
-----
125      15 CHARLY      SECONDO

begin
  dbms_resource_manager.switch_consumer_group_for_sess (
    session_id => 125,
    session_serial => 15,
    consumer_group => 'PRIMO');
end;
/

set lines 80
col sid format 999999
col serial# format 999999
col username format a8
```



```
col resource_consumer_group format a25
select sid, serial#, username, resource_consumer_group
from v$session where sid=125;
```

```
SID SERIAL# USERNAME RESOURCE_CONSUMER_GROUP
-----
125      15 CHARLY     PRIMO
```

```
set lines 120
COL name                                FORMAT A25      HEADING 'Resource|Consumer|Group'
COL active_sessions                    FORMAT 9999      HEADING 'Act|Sess'
COL execution_waiters                  FORMAT 9999      HEADING 'Exec|Wtrs'
COL requests                           FORMAT 9999      HEADING 'Reqs'
COL cpu_wait_time                      FORMAT 99999999 HEADING 'CPU|Wait|Time'
COL cpu_waits                          FORMAT 99999999 HEADING 'CPU|Waits'
COL consumed_cpu_time                  FORMAT 99999999 HEADING 'CPU|Time|Used'
COL yields                             FORMAT 9999      HEADING 'Ylds'
COL queue_length                       FORMAT 999999   HEADING 'Queue|Len'
COL current_undo_consumption           FORMAT 999999   HEADING 'Curr|UNDO|Used'
```

```
SELECT
  name
,active_sessions
,execution_waiters
,requests
,cpu_wait_time
,cpu_waits
,consumed_cpu_time
,yields
,queue_length
,current_undo_consumption
FROM v$rsrc_consumer_group;
```

Resource Consumer Group	Act Sess	Exec Wtrs	Reqs	CPU Wait Time	CPU Waits	CPU Time Used	Ylds	Queue Len	Curr UNDO Used
PRIMO	0	0	0	0	0	1	0	0	0
SECONDO	1	0	8	0	0	216	0	0	0
_ORACLE_BACKGROUND_GROUP_	21	0	28	0	0	0	0	0	0

```
col name format a20
select n.name, s.value
from v$sesstat s,
     v$statname n
where s.statistic#=n.statistic#
and name like 'scheduler%'
and s.sid=125
order by value desc;
```

Resource Consumer Group	VALUE
scheduler wait time	0



Afficher max utilization limit.

La procédure ci-dessous vous permet de générer de la CPU dans votre session CHARLY.

```
declare
  x number;
begin
  for i in 1..20000000 loop
    x:=dbms_random.random;
  end loop;
end;
/
```

Les requêtes ci-dessous vous permettent d'observer l'utilisation de ressource manager depuis votre session connectée en SYS:

```
set lines 120
COL name                                FORMAT A25      HEADING 'Resource|Consumer|Group'
COL active_sessions                    FORMAT 9999      HEADING 'Act|Sess'
COL execution_waiters                  FORMAT 9999      HEADING 'Exec|Wtrs'
COL requests                           FORMAT 9999      HEADING 'Reqs'
COL cpu_wait_time                      FORMAT 99999999  HEADING 'CPU|Wait|Time'
COL cpu_waits                          FORMAT 99999999  HEADING 'CPU|Waits'
COL consumed_cpu_time                  FORMAT 99999999  HEADING 'CPU|Time|Used'
COL yields                             FORMAT 9999      HEADING 'Ylds'
COL queue_length                       FORMAT 99999     HEADING 'Queue|Len'
COL current_undo_consumption           FORMAT 99999     HEADING 'Curr|UNDO|Used'
```

```
SELECT
  name
,active_sessions
,execution_waiters
,requests
,cpu_wait_time
,cpu_waits
,consumed_cpu_time
,yields
,queue_length
,current_undo_consumption
FROM v$rsrc_consumer_group;
```

Resource Consumer Group	Act Sess	Exec Wtrs	Reqs	CPU Wait Time	CPU Waits	CPU Time Used	Ylds	Queue Len	Curr UNDO Used
PRIMO	0	0	0	58467	92	31664	92	0	0
SECONDO	1	0	9	0	0	246	0	0	0
_ORACLE_BACKGROUND_GROUP_	21	0	29	0	0	0	0	0	0

```
col name format a20
select n.name, s.value
  from v$sesstat s,
       v$statname n
 where s.statistic#=n.statistic#
       and name like 'scheduler%'
       and s.sid=125
 order by value desc;
```

Group	VALUE
scheduler wait time	5987

```
select begin_time, consumer_group_name, cpu_consumed_time, cpu_wait_time
  from v$rsrcmgrmetric_history
```



```
where consumer_group_name='PRIMO'  
order by begin_time;
```

BEGIN_TIM	CONSUMER_GROUP_NAME	CPU_CONSUMED_TIME	CPU Wait Time
21-FEB-10	PRIMO	0	0
21-FEB-10	PRIMO	0	0
21-FEB-10	PRIMO	0	0
21-FEB-10	PRIMO	9082	16314
21-FEB-10	PRIMO	20354	38386
21-FEB-10	PRIMO	2227	3767
21-FEB-10	PRIMO	0	0

Désactiver et supprimer le plan de ressource

```
alter system set resource_manager_plan='';  
  
begin  
  DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;  
  DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;  
  
  DBMS_RESOURCE_MANAGER.DELETE_PLAN (  
    plan => 'LIMITED_CPU');  
  DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP (  
    consumer_group=>'PRIMO');  
  
  DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;  
  DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;  
end;  
/
```



23.4 La gestion des droits

Dans une base Oracle, les droits des utilisateurs sont gérés avec la notion de privilège.

Un privilège est le droit :

- ♦ D'exécuter un ordre SQL en général (par exemple, créer une table) ⇒ **Privilège système** D'accéder à un objet d'un autre utilisateur (par exemple, mettre à jour les données de la table CLIENT) ⇒ **Privilège objet**

Les privilèges peuvent être attribués directement aux utilisateurs ou par l'intermédiaire de rôles :

⇒ Un rôle est un regroupement nommé de privilèges

23.4.1 Privilèges systèmes

Chaque ordre SQL a au moins un privilège système associé. Souvent, l'ordre SQL a un privilège système qui porte le même nom.

Par exemple, l'ordre CREATE TABLE a un privilège système associé qui s'appelle CREATE TABLE (donne le droit de créer une table dans son propre schéma). Certains privilèges système reprennent le nom de l'ordre SQL avec le mot clé ANY. Dans ce cas, le privilège système permet d'exécuter l'ordre dans n'importe quel schéma de la base.

Par exemple, le privilège système CREATE ANY TABLE donne le droit de créer une table dans n'importe quel schéma de la base.

Les privilèges système doivent être attribués avec modération (notamment les privilèges ANY).

Attribuer un privilège système à un utilisateur

L'ordre SQL GRANT permet d'attribuer un privilège système.

```
GRANT nom_privilège [,...]
TO { nom_utilisateur | PUBLIC } [,...]
[ WITH ADMIN OPTION ] ;
```

```
| GRANT CREATE SESSION, CREATE TABLE TO clo;
```

Le privilège peut être attribué à un utilisateur ou à tous les utilisateurs (PUBLIC).

La clause WITH ADMIN OPTION donne au bénéficiaire le droit de transmettre le privilège système.

Le privilège attribué est immédiatement actif.

Pour attribuer un privilège système, il faut avoir reçu :

⇒ Le privilège en question avec la clause WITH ADMIN OPTION Ou le privilège système GRANT ANY PRIVILEGE





L'ordre GRANT ALL PRIVILEGES TO, attribue d'un seul coup TOUS les privilèges système à un utilisateur.

Enlever un privilège système à un utilisateur

L'ordre SQL REVOKE permet d'enlever un privilège système

```
REVOKE nom_privilège [,...]
FROM { nom_utilisateur | PUBLIC } [,...] ;
```

```
REVOKE CREATE TABLE FROM clo;
```

Le privilège est immédiatement enlevé et ne peut plus être exercé.

Pour enlever un privilège système, il faut avoir reçu :

- ⇒ Le privilège en question avec la clause WITH ADMIN OPTION Ou le privilège système GRANT ANY PRIVILEGE

Il n'y a pas de cascade dans la révocation d'un privilège système qui a été transmis grâce à la clause WITH ADMIN OPTION.

L'ordre GRANT ALL PRIVILEGES TO, attribue d'un seul coup TOUS les privilèges système à un utilisateur. A manipuler avec précaution.



L'ordre REVOKE ALL PRIVILEGES FROM, enlève d'un seul coup TOUS les privilèges système à un utilisateur.

Les privilèges SYSDBA et SYSOPER Nous avons déjà vu que les privilèges SYSDBA et SYSOPER étaient nécessaires pour réaliser certaines opérations d'administration (démarrage/arrêt/création de base de données).

Ces privilèges peuvent être contrôlés, soit par une appartenance à un groupe particulier du système d'exploitation, soit par un fichier de mots de passe. Dans le cas de l'utilisation d'un fichier de mots de passe, par défaut, seul SYS a reçu les privilèges SYSDBA et SYSOPER. Si le fichier de mots de passe est exclusivement associé à la base (paramètre REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE), il est possible d'attribuer l'un ou l'autre de ces privilèges à d'autres utilisateurs.

L'attribution et la révocation s'effectuent avec les ordres SQL GRANT et REVOKE :

- ⇒ Pour attribuer le privilège SYSDBA, il faut être connecté AS SYSDBA.
- ⇒ Pour attribuer le privilège SYSOPER, il faut être connecté, AS SYSOPER.



L'attribution de ces privilèges à d'autres utilisateurs n'est possible que si la base fonctionne avec un fichier de mots de passe en mode EXCLUSIF.

La vue V\$PWFILERS, permet de lister les utilisateurs qui ont reçu le privilège SYSDBA ou SYSOPER (cette vue est toujours vide si REMOTE_LOGIN_PASSWORDFILE=NONE).

23.4.2 Privilèges objets

Par défaut, seul le propriétaire d'un objet a le droit d'y accéder.

Pour qu'un autre utilisateur puisse accéder à l'objet, le propriétaire doit lui donner un des privilèges objet suivants.



Avoir un droit sur un objet ne dispense pas de préfixer le nom de l'objet par le nom du schéma de l'objet.

Pour faciliter l'écriture des requêtes et rendre le schéma de l'objet transparent, il faut utiliser des synonymes (voire des synonymes publics).

Réciproquement, l'existence d'un synonyme même public, ne donne aucun droit sur l'objet sous-jacent.

Ces privilèges donnent les droits suivants :

- ♦ select : droit de lecture des données
- ♦ insert : droit de création de ligne
- ♦ update : droit de modification de ligne
- ♦ delete : droit de suppression de ligne
- ♦ execute : droit d'exécuter un programme de type procédure, fonction, package
- ♦ alter : droit de modifier un objet
- ♦ index : droit de créer un index sur une table
- ♦ reference : droit de créer une contrainte de clé étrangère sur une table

Les privilèges INSERT, UPDATE et REFERENCES peuvent être restreints à certaines colonnes.

La liste des privilèges objets est visualisable dans la vue

Les principaux privilèges objets sont présentés dans le tableau ci-dessous.



Privilège	Table	Vue	Séquence	procédures
SELECT	✓	✓	✓	
INSERT	✓	✓		
UPDATE	✓	✓		
DELETE	✓	✓		
EXECUTE				✓
ALTER	✓		✓	
INDEX	✓			
REFERENCES	✓			

Attribuer un privilège objet à un utilisateur

L'ordre SQL GRANT permet d'attribuer un privilège objet.

```
GRANT { nom privilège[(liste colonnes)] [,...] | ALL [PRIVILEGES] }  
ON [nom schéma.]nom objet  
TO { nom utilisateur | PUBLIC } [,...]  
[ WITH GRANT OPTION ] ;
```

- Pour les privilèges INSERT, UPDATE ou REFERENCES, une liste de colonnes peut être spécifiée
- Le mot clé ALL permet d'attribuer tous les privilèges
- Le privilège peut être attribué à un utilisateur ou à tous les utilisateurs (PUBLIC)
- La clause WITH GRANT OPTION donne au bénéficiaire le droit de transmettre le privilège objet

```
| GRANT SELECT, INSERT, UPDATE(nom,prenom) ON betty.adherent TO charly;
```

Le privilège attribué est immédiatement actif.

Pour attribuer un privilège objet, il faut :

- ⇒ Être propriétaire de l'objet Ou avoir reçu le privilège en question avec la clause WITH GRANT OPTION



Certains privilèges système donnent implicitement des privilèges objet sur tous les objets

- ♦ ALTER ANY SEQUENCE
- ♦ ALTER ANY TABLE
- ♦ CREATE ANY INDEX
- ♦ DELETE ANY TABLE
- ♦ EXECUTE ANY PROCEDURE
- ♦ INSERT ANY TABLE
- ♦ SELECT ANY SEQUENCE
- ♦ SELECT ANY TABLE
- ♦ UPDATE ANY TABLE

Pour développer des vues ou des programmes PL/SQL utilisant les objets d'un autre schéma, il faut avoir reçu les privilèges objet nécessaires en direct et non à travers un rôle.

Utiliser des synonymes publics permet de simplifier l'écriture des requêtes et de rendre transparent le propriétaire des objets.

Il est possible d'indiquer quel schéma utiliser en premier pour la résolution des noms dans une session par l'ordre SQL : ALTER SESSION SET CURRENT_SCHEMA (par défaut c'est le schéma de l'utilisateur). Comme le synonyme cet ordre ne donne pas le droit sur les objets du schéma, il faut avoir reçu les privilèges nécessaires.

Par défaut un utilisateur qui a un privilège EXECUTE sur une procédure stockée n'a pas besoin d'avoir les privilèges sur les objets manipulés par la procédure.

Il existe un package DBMS_RLS, qui permet de mettre en place des mécanismes de filtres (lectures et mises à jour) sur les lignes des tables (*Oracle8i Supplied PL/SQL Package Reference*).

Depuis la version 8i, il est possible de choisir le mode souhaité pour l'exécution des programmes (*Oracle8i Supplied PL/SQL Package Reference*) :

- ⇒ Avec les droits du propriétaire (clause AUTHID DEFINER dans la création du programme) : c'est l'ancienne méthode.
- ⇒ Avec les droits de l'appelant (clause AUTHID CURRENT_USER dans la création du programme) : c'est une nouvelle possibilité qui présente d'autres avantages.

Enlever un privilège objet à un utilisateur

L'ordre SQL REVOKE permet d'enlever un privilège objet :

```
REVOKE { nom privilège [,...] | ALL [PRIVILEGES] }  
ON [nom schéma.]nom objet  
FROM { nom utilisateur | PUBLIC } [,...]  
[ CASCADE CONSTRAINTS ] ;
```

Lors de la révocation du privilège REFERENCES, la clause CASCADE CONSTRAINTS doit être présente si des contraintes d'intégrité référentielle ont été créées grâce à ce privilège.

```
| REVOKE INSERT, UPDATE TABLE ON client FROM clo;
```



Le privilège est immédiatement enlevé et ne peut plus être exercé.

Pour enlever un privilège objet, il faut :

- ⇒ Être propriétaire de l'objet Ou avoir reçu le privilège en question avec la clause WITH GRANT OPTION



L'ordre REVOKE ALL ON ... FROM ..., enlève d'un seul coup TOUS les privilèges objet à un utilisateur.

Lorsque que l'on enlève un privilège objet à un utilisateur, le privilège est immédiatement enlevé et ne peut plus être exercé.

Il n'est pas possible d'enlever à PUBLIC un privilège en pensant l'enlever à tout le monde.

Il y a cascade dans la révocation d'un privilège objet qui a été transmis grâce à la clause WITH GRANT OPTION.

Si le privilège est retiré à Pierre, et que Pierre l'a transmis à Paul, alors Paul ne l'a plus.

23.5 Les Rôles

Un rôle est un regroupement nommé de privilèges (système et objet) qui peut être attribué à un utilisateur. Tous les privilèges regroupés dans le rôle sont alors simultanément attribués à l'utilisateur.

- ♦ Un rôle peut être attribué à un autre rôle.
- ♦ Un utilisateur peut avoir plusieurs rôles.
- ♦ Un rôle n'appartient à personne.

La mise en œuvre s'effectue en trois étapes :

- ⇒ Création du rôle
- ⇒ Attribution des privilèges (système et objet) au rôle
- ⇒ Attribution du rôle aux utilisateurs

23.5.1 Création d'un rôle

L'ordre SQL CREATE ROLE permet de créer un rôle :

```
CREATE ROLE nom  
[ IDENTIFIED { BY mot_de_passe | EXTERNALLY } | NOT IDENTIFIED ] ;
```

- IDENTIFIED BY mot_de_passe : Indique qu'un mot de passe est nécessaire pour activer le rôle
- IDENTIFIED EXTERNALLY : Indique qu'une identification externe (système d'exploitation par exemple) est nécessaire pour activer le rôle
- NOT IDENTIFIED (valeur par défaut) : Indique qu'aucune identification n'est nécessaire pour activer le rôle



```
| CREATE ROLE stage;
```

23.5.2 Attribuer des privilèges à un rôle

L'ordre SQL GRANT permet d'attribuer des privilèges système ou des privilèges objet à un rôle :

Syntaxe pour les privilèges système

```
| GRANT nom_privilège [,...]
| TO nom_rôle [,...]
| [ WITH ADMIN OPTION ] ;
```

Syntaxe pour les privilèges objet

```
| GRANT { nom_privilège[(liste_colonnes)] [,...] | ALL [PRIVILEGES] }
| ON [nom schéma.]nom_objet
| TO nom_rôle [,...] ;
```

```
| GRANT CREATE SESSION, CREATE TABLE TO stage;
| GRANT SELECT, INSERT ON adherent TO stage;
```

C'est la même syntaxe que pour l'attribution directe à un utilisateur :

- ⇒ A l'exception de la clause WITH GRANT OPTION qui n'est pas permise pour l'attribution d'un privilège objet à un rôle

Les privilèges attribués sont immédiatement actifs pour les utilisateurs connectés qui ont le rôle actif.

Tout utilisateur a le droit d'attribuer un privilège à un rôle, du moment que l'utilisateur a le droit d'attribuer le privilège.

- ⇒ Rappel : le rôle n'appartient à personne

23.5.3 Enlever des privilèges à un rôle

L'ordre SQL REVOKE permet d'enlever des privilèges système ou des privilèges objet à un rôle :

Syntaxe pour les privilèges système

```
| REVOKE nom_privilège [,...]
| FROM nom_rôle [,...] ;
```

Syntaxe pour les privilèges objet

```
| REVOKE { nom_privilège [,...] | ALL [PRIVILEGES] }
| ON [nom schéma.]nom_objet
| FROM nom_rôle [,...] ;
```

```
| REVOKE CREATE TABLE FROM mailing;
| REVOKE UPDATE ON adherent FROM mailing;
```



C'est la même syntaxe que pour la révocation directe d'un utilisateur. Les privilèges sont immédiatement enlevés et ne peuvent plus être exercés par les utilisateurs connectés qui ont le rôle actif.

Tout utilisateur a le droit d'enlever un privilège à un rôle, du moment que l'utilisateur a le droit d'enlever le privilège.

⇒ Puisque le rôle n'appartient à personne

D'une manière générale, l'ordre REVOKE ne permet d'enlever à une « cible » (utilisateur, rôle ou PUBLIC) que ce qui a été attribué explicitement à cette « cible ».

Par exemple, si un privilège a été attribué à un rôle, il n'est pas possible de l'enlever directement à un utilisateur qui a reçu le rôle.

23.5.4 Attribuer un rôle à un utilisateur ou à un autre rôle

L'ordre SQL GRANT permet d'attribuer un rôle :

```
GRANT nom_rôle [,...]
TO { nom_utilisateur | PUBLIC | nom_rôle } [,...]
[ WITH ADMIN OPTION ] ;
```

Même syntaxe que pour l'attribution d'un privilège système

```
GRANT mailing TO clo;
```

Pour attribuer un rôle, il faut avoir reçu le rôle en question avec la clause WITH ADMIN OPTION ou le privilège système GRANT ANY ROLE.

Le créateur d'un rôle n'est pas propriétaire du rôle mais le rôle lui est attribué avec l'option WITH ADMIN OPTION dans ce cas il peut l'attribuer à son tour. Le rôle attribué n'est pas immédiatement actif.

23.5.5 Enlever un rôle à un utilisateur ou à un rôle

L'ordre SQL REVOKE permet d'enlever un rôle

```
REVOKE nom_rôle [,...]
FROM { nom_utilisateur | PUBLIC | nom_rôle } [,...] ;
```

Mêmes syntaxes que pour la révocation d'un privilège système

```
REVOKE mailing FROM clo;
```

Pour enlever un rôle, il faut avoir reçu le rôle en question avec la clause WITH ADMIN OPTION ou le privilège système GRANT ANY ROLE.

Lorsqu'un rôle est enlevé, les utilisateurs connectés avec le rôle actif peuvent continuer à exercer les privilèges associés jusqu'à la fin de la session ou la désactivation du rôle.



Lorsque le rôle est enlevé, les utilisateurs connectés peuvent toujours exercer les privilèges associés à ce rôle jusqu'à la fin de la session, ou la désactivation du rôle.



23.5.6 Supprimer un rôle

L'ordre SQL DROP ROLE permet de supprimer un rôle

```
DROP ROLE nom_rôle ;  
DROP ROLE mailing;
```

Pour supprimer un rôle, il faut avoir reçu le rôle en question avec la clause WITH ADMIN OPTION ou le privilège système DROP ANY ROLE.

Le rôle est immédiatement enlevé des utilisateurs.

Les privilèges associés ne peuvent immédiatement plus être exercés.

23.5.7 Activer ou désactiver un rôle

Un rôle attribué à un utilisateur (directement ou via un autre rôle) est par défaut automatiquement activé lors de la connexion de l'utilisateur.

Si l'utilisateur est connecté au moment de l'attribution, l'activation immédiate n'est pas automatique, mais l'utilisateur peut l'activer par l'ordre SQL SET ROLE.

Parmi les rôles attribués à l'utilisateur, il est possible de définir ceux qui sont effectivement automatiquement activés lors de la connexion de l'utilisateur.

- ➡ Ce sont les rôles par défaut définis par un ordre SQL ALTER USER L'utilisateur peut activer les autres par un ordre SQL SET ROLE

Les rôles doivent avoir été attribués à un utilisateur, il n'est pas possible de « s'auto attribuer » un rôle en l'activant.

Utiliser plusieurs rôles sans qu'ils soient tous actifs présente deux intérêts :

- ♦ Le paramètre MAX_ENABLED_ROLES (20 par défaut) limite le nombre de rôles actifs simultanément pour un utilisateur. Si un utilisateur utilise un nombre de rôles supérieur à cette limite, il est possible d'en désactiver certains pour en activer d'autres.
- ♦ Des rôles protégés par un mot de passe peuvent être attribués à des utilisateurs, mais inactifs par défaut et sans donner le mot de passe à l'utilisateur. C'est l'applicatif qui active le rôle en fournissant le mot de passe de façon transparente. Par exemple l'application se connecte et lorsque l'utilisateur se signe le rôle devient actif.



23.5.8 Définir des rôles par défaut

L'ordre SQL ALTER USER permet de définir les rôles par défaut d'un utilisateur

```
ALTER USER nom_utilisateur DEFAULT ROLE  
{ nom_rôle [,...] | ALL [ EXCEPT nom_rôle [,...] ] | NONE } ;
```

- ALL : Tous les rôles attribués à l'utilisateur sont activés par défaut, La clause EXCEPT permet d'en enlever certains.
- NONE : Aucun des rôles attribués à l'utilisateur n'est activé par défaut

```
ALTER USER oheu DEFAULT ROLE mailing;  
ALTER USER vdep DEFAULT ROLE ALL EXCEPT mailing;
```

L'ordre SQL SET ROLE permet d'activer ou de désactiver un rôle

```
SET ROLE  
{ nom_rôle [ IDENTIFIED BY mot_de_passe ] [,...]  
| ALL [ EXCEPT nom_rôle [,...] ] | NONE } ;
```

- IDENTIFIED BY : Donne le mot de passe qui permet d'activer le rôle
- ALL : Active tous les rôles attribués à l'utilisateur
- La clause EXCEPT permet d'en enlever certains
- NONE : Désactive tous les rôles

```
• L'utilisateur VDEP active le rôle MAILING  
SET ROLE mailing;
```

23.5.9 Rôles prédéfinis

Oracle propose une douzaine de rôles prédéfinis dont :

- ♦ **Connect** : permet la connexion d'un utilisateur
- ♦ **Resource** : permet la création des principaux objets d'un schéma (table, vues, ...)
- ♦ **Dbu** : donne tous les privilèges système avec la clause WITH ADMIN OPTION
- ♦ **Exp_full_database** : permet l'export complet de la base
- ♦ **Imp_full_database** : permet l'import complet de la base
- ♦ **Select_catalog_role** : permet de lire le dictionnaire de données (accéder aux vues DBA_ et V\$)
- ♦ **Execute_catalog_role** : permet d'exécuter les packages du dictionnaire de données
- ♦ **Delete_catalog_role** : permet de supprimer dans les tables du dictionnaire de données



23.6 Supervision des utilisateurs connectés

Identifier les utilisateurs actuellement connectés en utilisant la vue V\$SESSION.

```
SQL> SELECT sid,serial#,username,osuser,status FROM v$session;
```

SID	SERIAL#	USERNAME	OSUSER	STATUS
1	3		SYSTEM	ACTIVE
2	1		SYSTEM	ACTIVE
3	1		SYSTEM	ACTIVE
4	1		SYSTEM	ACTIVE
5	1		SYSTEM	ACTIVE
6	1		SYSTEM	ACTIVE
...				
10	10494	VDEP	vdep	INACTIVE
14	450	SYSTEM	Administrateur	ACTIVE

23.7 Déconnecter un utilisateur

Utiliser la commande ALTER SYSTEM DISCONNECT SESSION :

```
ALTER SYSTEM DISCONNECT SESSION 'sid,serial#'  
[ IMMEDIATE | POST_TRANSACTION ]  
;
```

```
SQL> ALTER SYSTEM DISCONNECT SESSION '9,24' IMMEDIATE;  
Système modifié.
```



L'utilisateur n'est prévenu de sa déconnexion à la base de données !

23.8 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les utilisateurs :

- DBA_SYS_PRIVS : privilèges système attribués aux utilisateurs ou aux rôles
- DBA_TAB_PRIVS : privilèges objet attribués aux utilisateurs ou aux rôles
- DBA_COL_PRIVS : colonnes concernées par les privilèges objet attribués aux utilisateurs ou aux rôles
- DBA_ROLE_PRIVS : rôles attribués aux utilisateurs ou aux rôles
- DBA_ROLES : liste des rôles existant dans la base
- ROLE_SYS_PRIVS : privilèges système attribués aux rôles (uniquement pour les rôles auxquels l'utilisateur a accès)
- ROLE_TAB_PRIVS : privilèges objet attribués aux rôles (uniquement pour les rôles auxquels l'utilisateur a accès)
- ROLE_ROLE_PRIVS : rôles attribués à d'autres rôles (uniquement pour les rôles auxquels l'utilisateur a accès)



- SESSION_PRIVS : privilèges système actuellement actifs dans la session (obtenus directement ou *via* un rôle)
- SESSION_ROLES : rôles actuellement actifs dans la session

DBA_ SYS_PRIVS	
GRANTEE	Nom de l'utilisateur ou du rôle qui a reçu le privilège système
PRIVILEGE	Privilège système reçu
ADMIN_OPTION	Privilège reçu avec la clause WITH ADMIN OPTION (yes, no)

DBA_ TAB_PRIVS	
GRANTEE	Nom de l'utilisateur ou du rôle qui a reçu le privilège objet
OWNER	Nom de l'utilisateur propriétaire de l'objet
TABLE_NAME	Nom de l'objet (pas forcément une table)
GRANTOR	Nom de l'utilisateur qui a attribué le privilège
PRIVILEGE	Privilège objet reçu
GRANTABLE	Privilège reçu avec la clause WITH GRANT OPTION (yes, no)

DBA_ ROLE_PRIVS	
GRANTEE	Nom de l'utilisateur ou du rôle qui a reçu le rôle
GRANTED_ROLE	Nom du rôle reçu
ADMIN_OPTION	Rôle reçu avec la clause WITH ADMIN OPTION (yes, no)



24 Les Objets de stockage

Les Principaux types d'objets d'un schéma sont :

- ◆ tables
- ◆ index
- ◆ vues
- ◆ séquences
- ◆ directory
- ◆ synonymes
- ◆ procédures stockées (fonctions, procédures, packages et triggers)

Les tables et les index occupent de l'espace de stockage en dehors de leur définition dans le dictionnaire de données.

Cet espace de stockage doit être planifié correctement pour éviter les erreurs liées au manque d'espace dans les tablespaces ou les problèmes de performance.

Ce sont des segments, dont le stockage est :

- ⇒ Organisé en extents
- ⇒ Piloté par la clause `STORAGE`

L'organisation du stockage dans les blocs est importante.

Il existe d'autres types d'objets qui occupent de l'espace de stockage :

- ◆ Les **vues matérialisées** : possédant une structure similaire aux tables dont le contenu est périodiquement mis à jour à partir d'une requête SQL
- ◆ Les **Index Organised Table (IOT)** : tables organisées en index dont le stockage est organisé dans l'index de la clé primaire de la table
- ◆ **Clusters et Hash cluster** : structures qui permettent de stocker physiquement l'ensemble des tables fréquemment interrogées par des jointures
- ◆ **Tables et index partitionnés** : l'option de partitionnement permet de découper le stockage physique des tables et des index en morceaux plus petits appelés partitions



25 Les tables

Il existe plusieurs types d'objets tels que les tables pour stocker des données. Ces données peuvent être stockées dans les objets suivants :

- ⇒ *Table standard*, c'est la table par défaut et la forme la plus couramment utilisée pour stocker les données.
- ⇒ *Table partitionnées*, qui permet le développement d'applications évolutives car elle comprend plusieurs partitions pouvant être situées dans des tablespaces différents. Les partitions sont utiles pour de grandes tables qui peuvent être interrogées à l'aide de plusieurs processus simultanés.
- ⇒ *Index-organized tables (IOT)*, toute la table est stockée dans l'index de la clé primaire. Oracle n'a plus besoin d'accéder à une deuxième structure.
- ⇒ *Table clusterisées*, utilisées pour optimiser les jointures.

Les tables sont stockées dans des tablespaces.

```
-- =====
•      Table : EMPLOYE
-- =====
create table EMPLOYE
(
    ID_EMP          INTEGER          not null,
    NOM             VARCHAR2(30)     not null,
    SALAIRE         NUMBER(4)        not null,
    EMPLOI          VARCHAR2(20)     null ,
    EMP_ID_EMP      INTEGER          null ,
    constraint PK_EMPLOYE primary key (ID_EMP)
using index
tablespace INDX
)
TABLESPACE DATA
/

-- =====
•      Index : A_POUR_PATRON_FK
-- =====
create index A_POUR_PATRON_FK on EMPLOYE (EMP_ID_EMP asc)
tablespace INDX
/

-- =====
•      Index : CLES ETRANGERES
-- =====
alter table EMPLOYE
add constraint FK_EMPLOYE_A_POUR_PA_EMPLOYE foreign key (EMP_ID_EMP)
references EMPLOYE (ID_EMP)
/
```



25.1 Les contraintes d'intégrité

Les différents types de contraintes que l'on trouve sous Oracle sont :

- ◆ NOT NULL (NN) : spécifie qu'une colonne ne peut pas contenir de valeurs nulles
- ◆ UNIQUE (UK) : désigne une colonne ou une combinaison de colonnes comme unique
- ◆ PRIMARY KEY (PK) : désigne une colonne ou une combinaison de colonnes comme clé primaire de la table
- ◆ FOREIGN KEY (FK) : désigne une colonne ou une combinaison de colonnes comme la clé étrangère dans une contrainte d'intégrité référentielle
- ◆ CHECK (CK) : spécifie une condition que chaque ligne de la table doit remplir

Bien que les contraintes NN et CK ne requièrent pas directement l'attention de l'administrateur, les contraintes PK, FK et UK doivent être gérées pour assurer une disponibilité élevée et des niveaux de performances acceptables.

Une contrainte d'intégrité peut être dans l'un des états suivants :

- ⇒ DISABLE : désactivée
- ⇒ NOVALIDATE ENABLE : non validée activée (contrainte forcée, données incohérentes)
- ⇒ VALIDATE ENABLE : validée activée

25.1.1 Contraintes immédiates ou différées

Les contraintes non différées ou IMMEDIATE sont appliquées à la fin de chaque ordre LMD.

Une violation de contrainte entraîne l'annulation de la transaction.

- ⇒ Une contrainte définie comme IMMEDIATE ne peut pas être modifiée pour être appliquée à la fin de la transaction.
- ⇒ Les contraintes différées sont vérifiées seulement lors de la validation d'une transaction.



Si une violation de contrainte est détectée, la transaction est annulée.

Pour qu'une contrainte soit de type différé, il faut la déclarer à sa création :

- ◆ INITIALLY IMMEDIATE : elle doit fonctionner par défaut comme une contrainte IMMEDIATE sauf si elle est définie autrement de façon explicite
- ◆ INITIALLY DEFERRED : elle est forcée par défaut à la fin de la transaction

Les applications peuvent forcer la contrainte pour qu'elle fonctionne en différé ou en mode immédiate.



Ceci se fait en utilisant les commandes ALTER SESSION ou SET CONSTRAINT :

```
alter session
set constraint[s] = { immediate | deferred | default }
;
```

25.1.2 Créer des contraintes d'intégrité

Lors de la création de la table, une contrainte peut être créée en fin de déclaration :

```
constraint nom contrainte ]
{ primary key ( nom colonne [ , nom colonne ] ... )
  [ using index clause index ]
| unique ( nom colonne [ , nom colonne ] ... )
  [ using index clause index ]
| foreign key ( nom colonne [ , nom colonne ] ... )
  [ schema. ] table [ ( nom colonne ) ]
  [ on delete cascade ]
| check ( regle conditions )
}
[ not deferrable
| deferrable ( initially { immediate | deferred } ) ]
[ disable | enable [ validate | novalidate ] ]
```



Il est conseillé d'adopter une convention standard de nommage des contraintes d'intégrité.

Il est possible de créer les tables sans les contraintes d'intégrité puis de rajouter celles-ci par une mise à jour de table ultérieure en utilisant la commande :

```
ALTER TABLE Nom table ADD CONSTRAINT Nom contrainte
;
```

```
-- =====
•      Index : CLES ETRANGERES
-- =====
alter table EMPLOYE
add constraint FK_EMPLOYE_A_POUR_PA_EMPLOYE foreign key (EMP_ID_EMP)
references EMPLOYE (ID_EMP)
/

alter table EMPLOYE
add constraint EMPLOYE_NOM_UK
unique(nom)
/

alter table EMPLOYE
add constraint SALAIRE_EMPLOYE_CC
check (salaire >0);
/
```



25.1.3 Désactiver les contraintes d'intégrité

Il est possible de désactiver les contraintes par la commande :

```
alter table [ schema. ] nom_table ]
disable { constraint nom_contrainte | primary key | unique (nom_colonne [ ,
nom_colonne ] ... ) }
[ cascade ]
;
```

La commande doit être utilisée pour une contrainte, si une clé primaire ou une contrainte unique est désignée comme clé étrangère, utilisez le paramètre `CASCADE` pour désactiver la clé étrangère avant de désactiver la clé primaire ou la contrainte unique.

De la même façon, il est possible d'activer des contraintes désactivées.

25.2 Les colonnes virtuelles en 11g

Dans la version 11g apparaît la notion de colonnes virtuelles.

Ce sont des colonnes affichées lors des requêtes SQL et utilisées pour des ordres DML (*Data Manipulation Language*) et DDL (*Data Définition Language*).

La colonnes virtuelles sont utilisées pour faciliter les requêtes des DATA Warehouse.

Il est possible de les utiliser sur des partitions de tables.

```
Lors de la création de la table
-- =====
•      Table : EMPLOYE
-- =====
create table EMPLOYE
(
  ID_EMP      INTEGER          not null,
  NOM         VARCHAR2(30)     not null,
  SALAIRE     NUMBER(4)        not null,
  EMPLOI      VARCHAR2(20)     null ,
  NB_ANNEE    INTEGER          not null,
  EMP_ID_EMP  INTEGER          null ,
  ANCIENNETE  AS (SALAIRE * 0.0005 * NB_ANNEE)
  constraint PK_EMPLOYE primary key (ID_EMP)
  using index
  tablespace INDX
)
TABLESPACE DATA
/
```

Dans notre exemple la colonne virtuelle `ANCIENNETE` représente la prime d'ancienneté calculée pour le salarié. Cette colonne est affichée au moment de la requête SQL.

Pour les insertions ou modifications de données il faut utiliser l'option `DEFAULT`.

```
INSERT INTO EMPLOYE
VALUES (2, 5000, 5, DEFAULT);
```



Il est possible de modifier une colonne virtuelle par un ALTER TABLE;

```
ALTER TABLE EMPLOYE  
  ANCIENNETE AS (SALAIRE * 0.0007 * NB_ANNEE) ;
```

Les colonnes virtuelles doivent obligatoirement référencer des colonnes existant dans la table d'origine.

25.3 Le ROWID

Le ROWID est une colonne virtuelle présente dans chaque table qui donne un identifiant unique de chaque ligne.

Il peut être interrogé comme les autres colonnes de la table :

```
SQL> SELECT ROWID, nom, salaire FROM employe;  
ROWID          NOM          SALAIRE  
-----  
AAABcFAADAAAAKAAA Gaston          1700  
AAABcFAADAAAAKAAB Spirou          2000  
AAABcFAADAAAAKAAC Titeuf          1800  
AAABcFAADAAAAKAAD Mariline        2000
```

Il permet de localiser physiquement la ligne, c'est le moyen le plus rapide pour accéder à une ligne.

- ⇒ Il est utilisé en interne par Oracle dans les index.
- ⇒ Il ne change jamais, tant que la ligne n'est pas supprimée.

Le ROWID n'est pas directement compréhensible. Dans la structure interne du ROWID, Oracle possède toutes les informations nécessaires à la localisation physique de la ligne (fichier de données, numéro de bloc, position de bloc).

Le package DBMS_ROWID comporte plusieurs fonctions permettant d'extraire les différents composants du ROWID.

Fonctions couramment utilisées :

- ◆ ROWID_INFO : informations sur le rowid
- ◆ ROWID_OBJECT : renvoie l'identifiant objet pour un rowid
- ◆ ROWID_RELATIVE_FNO : renvoie le numéro de fichier relatif pour un rowid
- ◆ ROWID_BLOCK_NUMBER : renvoie le numéro de bloc pour un rowid
- ◆ ROWID_ROW_NUMBER : renvoie le numéro de ligne pour un rowid
- ◆ ROWID_TO_ABSOLUTE_FNO : renvoie le numéro de fichier absolu pour un rowid
- ◆ ROWID_TO_EXTENDED : change un rowid de : limité à étendu
- ◆ ROWID_TO_RESTRICTED : change un rowid de : étendu à limité



⇒ La requête suivante permet d'obtenir l'emplacement physique des lignes d'une table.

```
Select nom, rowid,  
Dbms_rowid.rowid_object(rowid)      as "objet",  
Dbms_rowid.rowid_relative_fno(rowid) as "fichier relatif",  
Dbms_rowid.rowid_block_number(rowid) as "block"  
From   opdef.employe ;
```

NOM	ROWID	objet	fichier relatif	block
-----	-----	-----	-----	-----
Gaston	AAABcFAADAAAAAKAAA	5893		3 10
Spirou	AAABcFAADAAAAAKAAB	5893		3 10
Titeuf	AAABcFAADAAAAAKAAC	5893		3 10
Mariline	AAABcFAADAAAAAKAAD	5893		3 10

25.4 Bloc Oracle et lignes de tables

Structure d'un bloc

Sa taille est définie à la création de la base (paramètre `DB_BLOCK_SIZE`).

L'en-tête du bloc contient l'adresse du bloc, le type de segment, un répertoire des tables, un répertoire des lignes et des entrées pour les transactions :

⇒ taille variable : en moyenne de l'ordre de 100 octets.

Le reste du bloc contient les données (une à plusieurs lignes de la table) et de l'espace libre.

L'en-tête est stocké dans la partie haute du bloc et les données sont insérées à partir du bas.

L'en-tête est susceptible de grossir (vers le bas) en fonction de l'activité dans le bloc, il ne rétrécit jamais.

Structure d'une ligne

L'en-tête de la ligne contient quelques informations (nombre de colonnes, chaînage éventuel, verrou)

⇒ Taille variable (3 octets minimum)

La taille de la ligne varie donc selon son contenu, c'est la somme de la taille des colonnes qui la constituent, sa structure est la suivante :

L'en-tête contient quelques informations sur la ligne (nombre de colonnes, chaînage éventuel, verrou), sa taille est variable (3 octets minimum).

⇒ Chaque colonne est stockée avec un en-tête (sur 1 à 3 octets = longueur de la colonne) et sa valeur

Structure d'une colonne

Chaque colonne est stockée avec un en-tête (qui donne la longueur de la colonne) et la valeur.

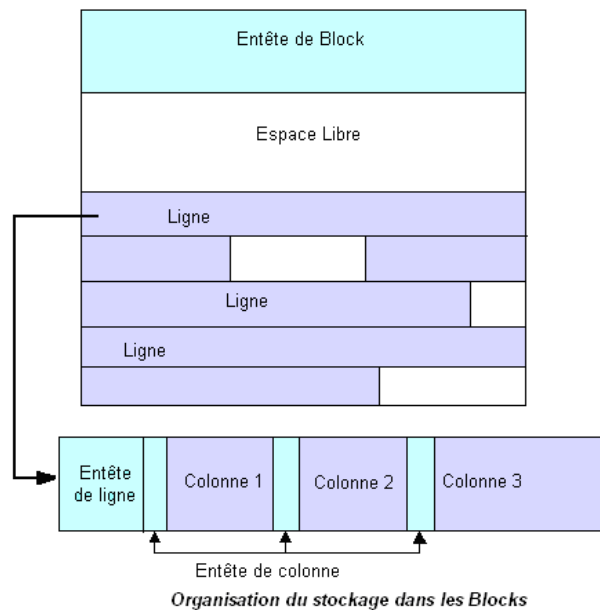
Taille variable de 1 à 3 octets

La longueur totale d'une colonne dépend du type de la colonne et de la valeur stockée dans la colonne.

Les différents types de données :



- ♦ `char (n)` = Longueur fixe (n octets), quelle que soit la valeur stockée dans la colonne
- ♦ `varchar2(n)` = Longueur variable (0 à n octets), dépendant de la valeur stockée dans la colonne
- ♦ `number (x,y)` = Longueur variable (entre 1 à 21 octets), dépendant de la valeur stockée dans la colonne
- ♦ `null` = 1 octet en milieu de ligne et aucun en fin de ligne
- ♦ `date` = Longueur fixe (8 octets)



Les fonctions SQL `VSIZE` et `DUMP` appliquées à une valeur (colonne, résultat d'une expression) permettent de connaître respectivement la taille en octets du stockage interne de la valeur et la représentation interne de la valeur.

25.5 La High Water Mark

Pour chaque table (et plus généralement segment), Oracle connaît le dernier bloc utilisé par la table, c'est la *High water mark* (HWM) = « ligne de plus hautes eaux ».

- ⇒ La HWM augmente lors des insertions ... mais ne diminue pas lors des suppressions.
- ⇒ Cela permet de connaître le nombre total maximum de blocs utilisés par la table dans toute son existence.

Lorsque la ligne est supprimée, Oracle ne réorganise pas le stockage des lignes à l'intérieur de la table. Il ne déplace pas une ligne déjà stockée à la fin de la table dans l'espace libéré, ce serait trop coûteux en temps de traitement et modifierait le `ROWID` de la ligne déplacée.



Lorsque Oracle fait un parcours complet de la table, il ne parcourt pas tous les blocs alloués à la table mais uniquement ceux situés sous la HWM.

Supprimer des lignes dans une table libère de l'espace utilisable par de futures lignes de la même table mais pas par des lignes d'autres tables.

25.6 Pilotage du remplissage d'un bloc

Deux paramètres de la définition de la table permettent de piloter le remplissage du bloc :

- ⇒ **PCTFREE** : pourcentage de l'espace du bloc laissé libre pour les modifications des lignes stockées dans le bloc
- ⇒ **PCTUSED** : pourcentage d'occupation auquel le bloc doit redescendre avant de redevenir candidat à l'insertion

La clause **PCTFREE** permet de ne pas remplir les blocs à 100% et de conserver de l'espace disponible à l'intérieur du bloc pour les futures mises à jour des lignes stockées dans le bloc.

Le paramètre **PCTUSED** permet de contrôler le moment où le bloc redeviendra candidat à l'insertion suite à la libération d'espace dans le bloc.

Ce paramètre permet d'éviter que le bloc ne redevienne immédiatement candidat à l'insertion dès que le moindre octet se libère, car l'espace libéré n'est peut-être pas suffisant pour réellement insérer une nouvelle ligne, de plus le bloc risque de redevenir non disponible à la première insertion.

Lorsqu'une ligne est modifiée, Oracle cherche à faire la modification en conservant la ligne à l'intérieur du bloc où elle est stockée.

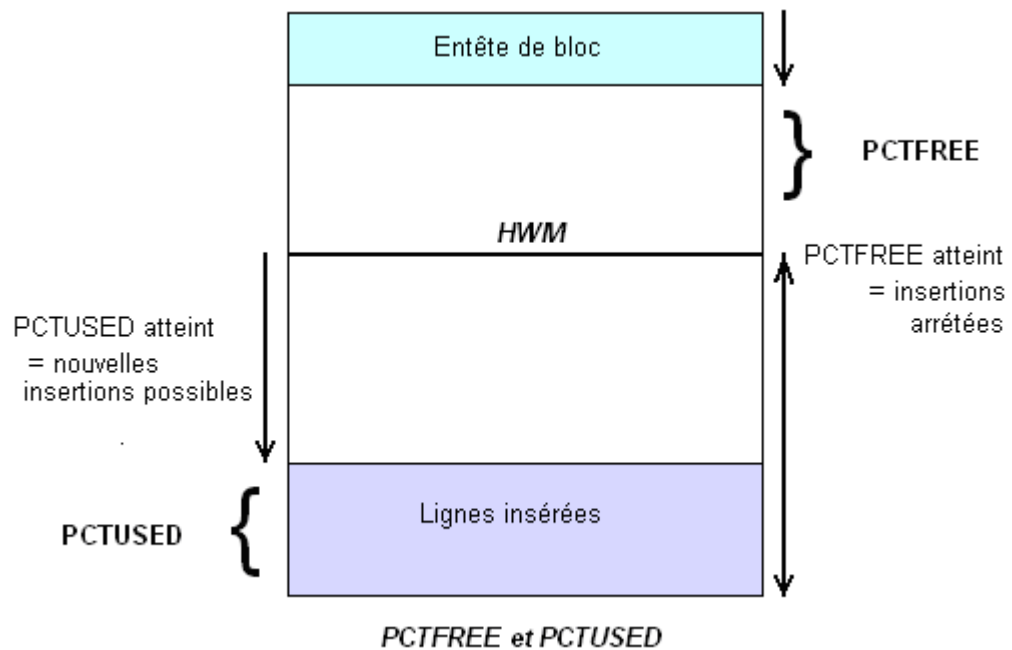
Si la longueur globale de la ligne diminue, cela ne pose pas de problème. Par contre, si la longueur globale de la ligne grandit et qu'il n'y a pas suffisamment de place disponible dans le bloc, Oracle déplace la ligne dans un autre bloc.

Lorsque le bloc atteint un taux de remplissage correspondant à **PCTFREE**, plus aucune insertion n'est réalisée.



La liste des blocs d'un segment disponible pour l'insertion de lignes s'appelle la **FREELIST**.





25.6.1 Calcul de PCTFREE et PCTUSED

PCTFREE :

$$\Rightarrow PCTFREE = 100 \times (1 - Ti / Tf)$$

- Ti = taille moyenne initiale de la ligne en octets (au moment de l'INSERT)
- Tf = taille moyenne finale de la ligne en octets (après UPDATE)

PCTUSED :

$$\Rightarrow PCTUSED = 100 - PCTFREE - Ps$$

- Ps étant le pourcentage de lignes à supprimer avant d'autoriser de nouvelles insertions dans le bloc (mettre un Ps assez élevé pour recevoir une ligne complète en INSERT)
- S'assurer que $Ps \times DB_BLOCK_SIZE$ est supérieur à la taille moyenne initiale d'une ligne (sinon le bloc sera candidat à l'insertion mais avec peu de chance d'avoir de l'espace disponible suffisant pour une ligne)



En règle générale, **PCTFREE + PCTUSED** est **compris entre 85 et 95%**.
Si la table se trouve dans un tablespace géré automatiquement par Oracle avec une **gestion automatique de segments**, alors la clause **PCTUSED** est **ignorée**.



25.6.2 Directives pour PCTFREE et PCTUSED

Pour une table de références ou ne faisant l'objet que d'INSERT, mettre un PCTFREE faible (0 à 5) et un PCTUSED élevé (99 à 95) pour un bon remplissage des blocs.

Pour une table faisant l'objet de beaucoup d'INSERT et d'UPDATE mais peu de DELETE, mettre un PCTFREE plus élevé (20 à 50 selon que les updates risquent ou non de faire grossir la ligne) et un PCTUSED élevé (80 à 90), espace réservé pour les mises à jour afin d'éviter les migrations.

Pour une table faisant l'objet de beaucoup d'INSERT, d'UPDATE et de DELETE, conservez un PCTFREE élevé (20 à 50) et mettre un PCTUSED relativement faible (60 à 80), espace réservé pour les mises à jour plus limitation des coûts de gestion vis à vis de la *freelist* (disponible/pas disponible).

25.7 Spécifier le stockage d'une table

Le stockage d'une table peut être spécifié lors de la création de la table (par CREATE TABLE) ou modifié ultérieurement (par ALTER TABLE) :

```
CREATE TABLE nom_table  
(spécification des colonnes)  
[ TABLESPACE nom_tablespace ]  
[ PARALLEL degrés ]  
[ PCTFREE valeur ]  
[ PCTUSED valeur ]  
[ clause stockage ]  
;
```

♦ clause stockage

```
STORAGE ( [ INITIAL valeur [K|M] ]  
[ NEXT valeur [K|M] ]  
[ MINEXTENTS valeur ]  
[ MAXEXTENTS { valeur | UNLIMITED } ]  
[ PCTINCREASE valeur ] )
```

- **TABLESPACE** = nom du tablespace dans lequel la table doit être créée
- **PARALLEL** = degrés de parallélisme utilisé par les requêtes SQL
- **PCTFREE** = valeur du PCTFREE (entre 0 et 100, 10 par défaut)
- **PCTUSED** = valeur du PCTUSED (entre 0 et 100, 40 par défaut)
- **STORAGE** = clause de stockage de la table
- **INITIAL** = taille du premier extent alloué
- **NEXT** = taille du deuxième extent alloué
- **MINEXTENTS** = nombre initial d'extent(s) alloué(s)
- **MAXEXTENTS** = nombre maximal d'extents allouables
- **PCTINCREASE** = pourcentage d'augmentation (0 à 100) de la taille des extents, à partir du troisième, par rapport au précédent



```
Lors de la création de la table
-- =====
•      Table : AVION
-- =====
create table AVION
(
    ID_AVION      INTEGER          not null,
    NOM_AVION     VARCHAR2(30)     null   ,
    constraint PK_AVION primary key (ID_AVION)
using index
tablespace INDX
)
TABLESPACE data
PARALLEL 4
PCTFREE 30
PCTUSED 60
STORAGE ( INITIAL 4000K NEXT 400K MAXEXTENTS 64) ;

Lors de la modification de la table
ALTER TABLE AVION
PCTFREE 40
PCTUSED 50
STORAGE ( NEXT 2000K ) ;
```



Si le nom du tablespace n'est pas présent, c'est le tablespace par défaut de l'utilisateur qui sera utilisé, ou le tablespace USERS défini comme tablespace de travail par défaut à la création de la base.

Les vues DBA_SEGMENTS et DBA_EXTENTS permettent de voir l'espace global alloué à la table

⇒ Ne donnent pas d'informations sur le nombre de blocs réellement utilisés

Attention à la gestion des fichiers gérés par Oracle qui adaptent la clause de stockage.

L'algorithme interne d'allocation des extents d'Oracle cherche à arrondir les extents à 5 blocs.

Dans le cas de tables accédées par index, Oracle réalise une entrée/sortie par bloc correspondant aux ROWID trouvés dans l'index.

Le paramètre DB_FILE_MULTIBLOCK_READ_COUNT correspond au nombre de blocs lus (DB_BLOCK_SIZE) en entrée/sortie par Oracle lors d'un parcours complet (FULL TABLE SCAN).

Exemple

Vous créez un tablespace géré localement à l'aide du script présenté ci-dessous :

```
Create tablespace DATA datafile
'c:/oracle/orcl/DATA01.dbf'
size 150M reuse
extent management local
uniform size 500K online
;
```

Après avoir créé le tablespace, vous créez une table contenant un segment initial de 100K et un minextent égal à 3 dans la clause de stockage.



Après création de la table, l'espace réservé pour celle-ci dans le tablespace est de :

⇒ 1 500K

25.8 Chaînage et migrations

En règle générale, une ligne de table est stockée complètement à l'intérieur d'un bloc :

⇒ Pour lire la ligne, Oracle n'a besoin de lire qu'un seul bloc.

Si la ligne est trop grande, Oracle la stocke dans plusieurs blocs chaînés par des pointeurs. C'est le phénomène de chaînage d'une ligne :

⇒ Pour lire cette ligne, Oracle a besoin de lire plusieurs blocs

Si une ligne grandit suite à une modification, et qu'il ne reste plus suffisamment d'espace dans le bloc, alors Oracle la déplace dans un autre bloc pointé par l'en-tête de la ligne resté dans le bloc d'origine :

- ⇒ C'est le phénomène de migration d'une ligne
- ⇒ Le ROWID de la ligne modifiée et migrée n'a pas changé
- ⇒ Mais pour lire cette ligne, Oracle a besoin de lire deux blocs

Le chaînage se produit lorsque le `DB_BLOCK_SIZE` est petit et qu'une table comporte un grand nombre de colonnes ou des colonnes stockant de grandes valeurs (type `LONG` ou `CLOB`).

Un `DB_BLOCK_SIZE` important (maximum 32 K) permet de limiter le chaînage mais pas forcément de le faire disparaître.

Le phénomène de chaînage est difficilement évitable :

- ♦ Sauf en augmentant la taille des blocs (y penser à la création de la base)
- ♦ Peut ne pas être suffisant pour les très grandes lignes

Le phénomène de migration peut être évité :

- ♦ En laissant suffisamment d'espace disponible dans les blocs pour les mises à jour
- ♦ Le paramètre `PCTFREE` doit donc être positionné avec soin sur les tables pour lesquelles la taille des lignes insérées est sensiblement inférieure à la taille des lignes après modification(s)

25.9 Rappel sur la génération des statistiques

Depuis la version 10g un job tourne en automatiquement de 22h à 6h les jours de la semaines et tout le week end, pour mettre à jour les statistiques des tables et des index de la base de données. C'est le job : `GATHER_STATS_JOB`.



Le job GATHER_STATS_JOBS lance une procédure interne :

- ⇒ DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC (similaire à la procédure DBMS_STATS.GATHER_DATABASE_STATS avec option GATHER_AUTO) mais en priorisant les objets de la base de données, c'est-à-dire les objets ayant le plus besoin de statistiques.

25.9.1 Statistiques pour les tables dans le dictionnaire de données

Les statistiques mettent à jour les colonnes de la vue du dictionnaire de données DBA_TABLES et ses consœurs.

DBA_TABLES	
✓	NUM_ROWS = Nombre de lignes de la table.
✓	BLOCKS = Nombre de blocs sous la HWM (blocs utilisés).
✓	EMPTY_BLOCKS = Nombre de blocs au dessus de la HWM (blocs inutilisés).
✓	AVG_SPACE = Espace libre moyen en octets dans les blocs occupés (sous la HWM).
✓	AVG_ROW_LEN = Longueur moyenne d'une ligne en octets, en tenant compte des informations de contrôle (en-tête de lignes et en-tête de colonnes).
✓	CHAIN_CNT = Nombre de lignes chaînées ou migrées.
✓	SAMPLE_SIZE = Taille de l'échantillon utilisé en cas d'analyse réalisée sur la table.
✓	LAST_ANALYZED = Date et heure de la dernière analyse réalisée sur la table.

La gestion manuelle de la collecte des STATISTIQUES est faite en utilisant le package DBMS_STATS.

Si vous êtes connecté comme utilisateur SYS, vous pouvez collecter manuellement des statistiques sur tous les objets de la base avec la commande :

```
EXEC DBMS_STATS.GATHER FIXED OBJECTS STATS('ALL') ;
```

Si vous êtes connecté comme utilisateur SYSTEM, vous pouvez collecter manuellement des statistiques sur les objets d'un schéma ou sur une table avec les commandes :

```
DBMS_STATS.GATHER TABLE STATS('schema','table') ;
```

```
DBMS_STATS.GATHER_SCHEMA_STATS('schema') ;
```

Exemples

```
Exec dbms_stats.gather_schema_stats('charly');  
Exec dbms_stats.gather_table_stats('charly','avion');
```



25.9.2 Interpréter les statistiques générées

Les problèmes pouvant être détectés sont l'espace inutilisé alloué à une table et le faible taux d'occupation des blocs.

La génération de statistiques alimente les colonnes de la vue `DBA_TABLES`.

La valeur de `BLOCKS` est toujours exacte même si les statistiques sont inexactes ou manquantes.

Espace utilisé par une table

Espace inutilisé alloué à une table :

- ⇒ Le nombre de blocs inutilisés alloués à la table (`EMPTY_BLOCKS`) est important et la table ne va plus grossir (ou peu)
- ⇒ Lié à une clause `STORAGE` mal adaptée

Faible taux d'occupation des blocs :

Le rapport $(DB_BLOCK_SIZE - AVG_SPACE) / DB_BLOCK_SIZE$ est faible et les lignes actuelles ne vont pas grossir et peu de nouvelles lignes vont être insérées :

- ⇒ Lié à des valeurs `PCTFREE` et `PCTUSED` mal adaptées ou à une suppression importante de données, pose un problème de performance dans le parcours complet de la table

25.9.3 Le package `DBMS_SPACE`

Le package `DBMS_SPACE` possède plusieurs procédures qui permettent de superviser le stockage d'un segment.

Les principales procédures sont :

- ◆ `FREE_BLOCKS` : informations sur les blocs libres dans un segment dont l'espace est géré manuellement.
- ◆ `SPACE_USAGE` : informations sur l'occupation des blocs dans un segment dont l'espace est géré automatiquement.
- ◆ `UNUSED_SPACE` : informations sur les blocs inutilisés d'un segment.

Ce package possède d'autres procédures qui permettent d'estimer la taille d'un segment (table ou index), ou la tendance de croissance de celui-ci.

```
•      liste des blocs vides et occupés dans la table AVION de CHARLY
Select t.blocks  Occupes, s.blocks  Alloues
From dba_tables t, dba_segments s
Where s.segment_name = t.table_name
And s.owner = t.owner
And t.table_name = 'AVION'
And t.owner = 'CHARLY'
;
```



25.10 Réorganiser le stockage d'une table

Différents besoins existent :

- ⇒ Améliorer le taux de remplissage des blocs
- ⇒ Corriger un problème de migration
- ⇒ Réorganiser plus globalement le stockage de la table (changement de tablespace, modification de la clause `STORAGE`, réorganisation immédiate avec de nouveaux paramètres `PCTFREE` ou `PCTUSED`, réduction du nombre d'extents alloués, ...)

Différentes techniques sont possibles :

- ◆ Recréer la table
- ◆ Export/Import
- ◆ Ordre SQL `ALTER TABLE ... MOVE`
- ◆ Ordre SQL `ALTER TABLE ... SHRINK SPACE`

Historiquement l'Import / Export est la technique de base pour les réorganisations un peu massives ; c'est d'ailleurs toujours la bonne technique pour la réorganisation complète de la base, notamment en cas de changement de la taille du bloc.



Depuis la version 8i, l'ordre SQL `ALTER TABLE ... MOVE` est le meilleur outil pour réorganiser une table.

25.10.1 L'ordre SQL Alter table ...Move

L'ordre SQL `ALTER TABLE ... MOVE`, permet de réorganiser complètement le stockage physique d'une table sans la supprimer :

```
ALTER TABLE nom table MOVE
[ TABLESPACE nom tablespace ]
[ PCTFREE valeur ]
[ PCTUSED valeur ]
[ clause stockage ]
;
```

- ◆ `clause_stockage`

```
STORAGE ( [ INITIAL valeur [K|M] ]
          [ NEXT valeur [K|M] ]
          [ MINEXTENTS valeur ]
          [ MAXEXTENTS { valeur | UNLIMITED } ]
          [ PCTINCREASE valeur ] )
```




```
ALTER TABLE employe MOVE  
PCTFREE 40  
PCTUSED 50  
/
```

Toutes les options de stockage peuvent être modifiées et les objets dépendants sont préservés.

Le principe est de recopier physiquement les données de la table dans de nouveaux extents, dans le même tablespace ou un autre tablespace. Il faut donc prévoir de l'espace disque.



Les index créés sur la table sont marqués UNUSABLE ; il faut les reconstruire (ALTER INDEX ... REBUILD).
La table n'est pas disponible en mise à jour, mais accessible en lecture seule.

25.10.2 L'ordre SQL : SHRINK

Avec les versions précédentes, l'espace une fois alloué en dessous de la HWM du segment ne pouvait être libéré que par la redéfinition ou le déplacement du segment.

Avec la version 10g, les segments peuvent être réorganisés sans « recopie des objets ».

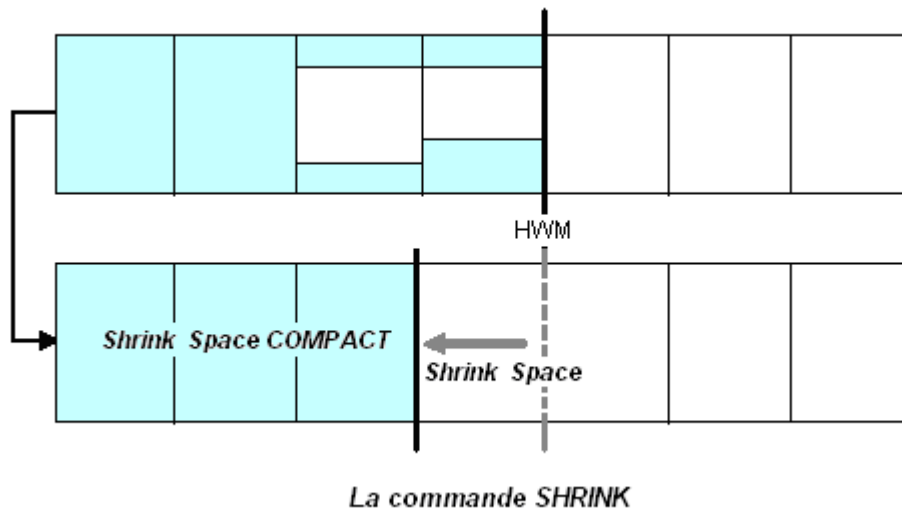
Cette fonctionnalité ne concerne que les objets contenus dans des tablespaces gérés par Oracle avec une gestion automatique des segments.

```
• exemple d'erreur  
ALTER TABLE charly.vol SHRINK SPACE CASCADE  
*  
ERROR at line 1:  
ORA-10635: Invalid segment or tablespace type  
  
• création d'un tablespace pour les tables  
• avec une gestion automatique des segments  
  
CREATE TABLESPACE DATA  
DATAFILE '&chemin\data01.dbf' SIZE 10M  
EXTENT MANAGEMENT LOCAL  
SEGMENT SPACE MANAGEMENT AUTO -- pour pouvoir faire les shrink  
/
```

Une réorganisation est composée de deux phases :

- ♦ La 1^{ère} phase consiste à compacter les données
Pendant cette phase, des lignes vont être déplacées vers la partie gauche du segment, autant que possible.
En interne, les lignes sont déplacées par paquets afin d'éviter des problèmes de LOCK.
Pour cela, la clause COMPACT n'est pas spécifiée.
- ♦ la 2^{ème} phase consiste à ajuster la HWM et l'espace inutilisé est libéré.
Si la clause COMPACT n'est pas spécifiée, l'espace du segment est compacté et, à la fin de la phase de compactage, la HWM est ajustée et l'espace inutilisé est libéré.





La partie supérieure du diagramme présente un segment avec des trous. Comme vous pouvez le constater, il y a de l'espace non utilisé au dessus et en dessous de la « *high-water mark* » (HWM) du segment.

La réorganisation de ce type de segment améliore les performances, ceci est vrai car il y a moins de blocs à lire en 1 seule fois quand le segment est réorganisé.

Ce qui est particulièrement vrai pour :

- ⇒ Le balayage entier de table « *full table scan* » (moins de blocs et blocs plus denses c'est à dire mieux remplis)
- ⇒ Un meilleur accès sur les index (moins d'opérations I/O pour le balayage des intervalles ROWID car les arbres sont plus compacts)



Une opération de réorganisation ne garantit pas que le nombre de lignes migrées soit réduit car cela ne concerne pas tous les blocs d'un segment, seulement ceux à réorganiser (les blocs pleins ne sont pas réorganisés, s'ils contiennent des lignes migrées, elles restent migrées).

Une opération de réorganisation s'effectue **ONLINE**, directement sur les données réelles, car elle ne demande pas d'espace supplémentaire pour être exécutée.



Une opération de réorganisation ne peut pas être exécutée sur des segments gérés par des « *free liste* ».

Les segments dans le tablespace « *Automatic Segment Space Management* » (ASSM) peuvent être réorganisés.



Même si certains objets stockés dans les tablespaces `ASSM` ne peuvent pas être réorganisés tels que :

- ◆ Les tables dans les clusters
- ◆ Les tables avec des colonnes de type `LONG`
- ◆ Les tables avec des vues matérialisées « on-commit »
- ◆ Les tables avec des vues matérialisées basées sur des `ROWID`
- ◆ Les segments `LOB`
- ◆ Les `IOT`
- ◆ Les segments *overflow* `IOT`
- ◆ Les tables contenant des index sur fonction



La dépendance des index est préservée pendant les opérations de réorganisation du segment. Les index sont en état d'utilisation après avoir réorganisé la table correspondante.

L'opération de réorganisation est traitée en interne comme une opération : *insert+delete* (très consommatrice en Redo Log).

Les triggers `DML` ne sont pas annulés si les données ne sont pas changées.

Les segments réorganisables sont :

- ⇒ Les tables organisées en *Heap* (pas d'index) ou avec des index
- ⇒ Quelques index
- ⇒ Partitions et les sous-partitions
- ⇒ Les vues matérialisées et leur log

Comme l'opération de réorganisation peut changer les `ROWID`, il faut permettre le déplacement des lignes dans le segment concerné, avant d'exécuter une telle opération.

Le déplacement des lignes est désactivé par défaut au niveau d'un segment.

- ⇒ Pour activer le déplacement des lignes la clause `ENABLE ROW MOVEMENT` du `CREATE TABLE` ou du `ALTER TABLE` est à utiliser.

```
SQL> -- attention le ts doit etre un ts
SQL> -- avec l'option SEGMENT SPACE MANAGEMENT AUTO
SQL> -- remise à ENABLE des ROW MOVEMENT
SQL> -- pour pouvoir faire le shrink après

SQL> alter table charly.vol ENABLE ROW MOVEMENT;
Table altered.
SQL> alter table charly.employe ENABLE ROW MOVEMENT;
```



La clause `SHRINK SPACE` est utilisée pour réorganiser l'espace dans un segment.

```
SQL> ALTER TABLE charly.employe SHRINK SPACE ;
Table altered.
SQL> ALTER TABLE charly.vol SHRINK SPACE ;
Table altered.
```

Pendant une opération de réorganisation, il est possible de demander à la base Oracle d'exécuter seulement la phase de compactage en spécifiant la clause `SHRINK SPACE COMPACT`.

```
ALTER TABLE charly.avion SHRINK SPACE COMPACT;
```

Si la clause `CASCADE` est spécifiée, la réorganisation sera effectuée pour tous les objets dépendants qui supportent une opération de réorganisation (segments).

```
ALTER TABLE charly.avion ENABLE ROW MOVEMENT ;
ALTER TABLE charly.avion SHRINK SPACE CASCADE ;
```

Remarques concernant l'exécution de SHRINK

La phase de compactage pour une opération de réorganisation se fait `ONLINE`. Ainsi, les opérations DML ainsi que les requêtes d'interrogation peuvent coexister pendant une opération de réorganisation.

Pendant la phase de compactage, les `LOCKs` sont gardés sur des lignes individuelles qui contiennent des données. Ceci permet de sérialiser les opérations DML concurrentes comme les `UPDATES` et les `DELETES`.

Pour éviter le `LOCK` de tout le segment, Oracle pose des `LOCKs` sur des paquets de lignes.

D'une façon similaire, les opérations DML conventionnelles peuvent bloquer l'avancement de la phase de compactage jusqu'à leur `COMMIT`.

La clause `COMPACT` est utilisée si vous avez des requêtes longues qui peuvent ralentir l'opération de réorganisation et qui essaient de lire à partir des blocs qui ont été sélectionnés.

Quand la clause `SHRINK SPACE COMPACT` est spécifiée, l'avancement de l'opération de réorganisation est sauvegardé dans les blocks bitmap du segment correspondant.

Ainsi pour la prochaine opération de réorganisation sur ce segment, la clause `SHRINK SPACE` pourra être réutilisée sans la clause `COMPACT` pendant la 2^{ème} phase car la base Oracle se souviendra de ce qui a déjà été fait.

Pendant la 2^{ème} phase de l'opération de réorganisation, quand la `HWM` est ajustée, l'objet est `LOCKE` en mode `EXCLUSIVE`. Cette phase dure très peu de temps et n'affecte pas la disponibilité du segment d'une manière significative, même si les « *curseurs* » qui en dépendent sont `INVALIDATED`.

25.11 Libérer de l'espace dans un segment

La clause `KEEP` permet de conserver un peu d'espace au-delà de la `HWM` sans préserver la taille initiale de la table.

L'ordre SQL `ALTER TABLE ... DEALLOCATE UNUSED` permet de libérer l'espace d'une table située au-dessus de la `HWM` :

```
ALTER TABLE nom table DEALLOCATE UNUSED [ KEEP valeur [K|M] ]
;
```



- **KEEP** : indique l'espace à conserver **au dessus** de la HWM
 - Oracle préserve la taille initiale du segment

```
ALTER TABLE employe DEALLOCATE UNUSED;  
• Oracle met à jour le dictionnaire de données et met la taille  
• de INITIAL à la valeur de la HWM.  
ALTER TABLE employe DEALLOCATE UNUSED KEEP 0;  
• Oracle met à jour le dictionnaire de données et met la taille de INITIAL à la  
valeur de 1 Mega octets.  
ALTER TABLE employe DEALLOCATE UNUSED KEEP 1M;
```

Cet ordre ne peut pas être utilisé pour libérer de l'espace au dessous de la HWM (espace libre suite à des suppressions de lignes).

L'espace libéré est rendu disponible pour d'autres segments visibles dans `DBA_FREE_SPACE`.

Fonctionnement

Sans clause `KEEP`, tout l'espace au dessus de la plus grande valeur (`HWM / taille initiale du segment`) est libéré.

Avec la clause `KEEP`, l'espace spécifié est conservé et les valeurs de `MINEXTENTS` et `INITIAL` sont éventuellement ajustées dans le dictionnaire.

Si le nombre d'extents est inférieur à `MINEXTENTS` alors `MINEXTENTS` est modifié dans le dictionnaire de données et devient égal au nombre d'extents.

Si la taille du premier extent devient inférieure à `INITIAL`, alors `INITIAL` est modifié dans le dictionnaire de données et devient égal à la taille du premier.

Un `KEEP 0` permet de ne pas conserver d'espace au-delà de la HWM et de ne pas préserver la taille initiale du segment : tout l'espace situé au-delà de la HWM est cette fois libéré.

25.12 Le monitoring d'une table

Il est possible de mettre une table sous surveillance en utilisant la clause `MONITORING/NOMONITORING` des ordres `CREATE TABLE` et `ALTER TABLE`.

Par défaut à partir de la version 10g, le monitoring des tables est activé.

Le mécanisme de supervision-modification est activé et les utilisateurs des procédures `GATHER AUTO` ou `STALE` de `DBMS_STATS` ne doivent plus activer explicitement le monitoring pour chaque table.

Vous pouvez toujours utiliser les clauses `[NO]MONITORING` dans les commandes `[CREATE | ALTER] TABLE` ainsi que les procédures `ALTER_SCHEMA_TAB_MONITORING` et `ALTER_DATABASE_TAB_MONITORING` du package `DBMS_STAT`.

Avec la version 10g, le paramètre d'initialisation `STATISTICS_LEVEL` fonctionne comme un *switch* global pour le mécanisme de supervision des tables plutôt que l'utilisation des clauses `[NO]MONITORING` dans les commandes `[CREATE | ALTER] TABLE`.

Il permet à l'optimiseur de détecter si la table a été accédée et si elle a besoin de statistiques.

- ♦ Syntaxe lors de la création d'une table

```
CREATE TABLE nom table (liste colonnes) MONITORING | NOMONITORING  
;
```

- ♦ Syntaxe lors de la modification d'une table

```
ALTER TABLE nom table MONITORING | NOMONITORING  
;
```



La colonne `MONITORING` de la vue `DBA_TABLES` indique si la table est sous surveillance

⇒ valeur `YES`

Par contre, si vous positionnez le paramètre `STATISTICS_LEVEL = BASIC`, la supervision est désactivée. S'il est égal à `TYPICAL` (valeur par défaut) ou `ALL`, la supervision (`monitoring`) est activée.

Les informations sur les tables surveillées peuvent être consultées dans la vue `DBA_TAB_MODIFICATIONS`.

25.13 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les tables :

- `DBA_TABLES` : informations sur les tables
- `DBA_TAB_COLUMNS` : informations sur les colonnes des tables
- `DBA_SEGMENTS` : informations sur les segments (dont ceux de type table)
- `DBA_EXTENTS` : informations sur les extents alloués à la table
- `DBA_TABLE_MODIFICATIONS` : informations sur les tables surveillées
- `DBA_OPTSTAT_OPERATIONS` : continent l'historique des opérations concernant les statistiques.
- `DBA_TAB_STATS_HISTORY` : contient l'historique des modification effectuées sur les tables.



26 Les index

Les tables et les contraintes créées, il faut créer des index sur les colonnes utilisées fréquemment en fonction du volume des tables et des requêtes, afin de réduire les temps de réponse.

Par défaut, Oracle crée un index unique sur les clés primaires.

Par contre, il faut créer des index sur les autres colonnes en fonction des besoins.



Si trop d'index sont créés sur une table, les temps de réponse s'alourdissent, surtout si les tables ont des volumes importants.

Les index réduisent les temps de réponse en lecture seule, mais pénalisent les performances en modification (INSERT, UPDATE, DELETE).

Un index peut comprendre une seule colonne (index unique), ou plusieurs colonnes (index concaténé).

Un index est donc un objet supplémentaire créé sur une ou plusieurs colonnes de table pour faciliter un accès rapide à ses données.

```
-- =====
•      Index: UTILISE_FK
-- =====
create index UTILISE_FK on VOL (ID_AVION asc)
tablespace INDX
/
```

26.1 Organisation logique

La classification logique d'un index dépend de la perspective de l'application.

Un index à colonne unique comprend une seule colonne dans la clé d'index.

Un *index concaténé*, également appelé *index composé*, est créé sur plusieurs colonnes d'une table. Les colonnes ne doivent ni suivre forcément le même ordre que celui des colonnes de la table, ni être adjacentes.

Un index comprend un maximum de 32 colonnes. Toutefois, la taille totale des colonnes reste inférieure à un tiers de la taille du bloc de données.

Un index unique garantit que deux lignes d'une table n'ont pas la même valeur dans la colonne qui le définit.



Dans un index non unique, plusieurs lignes de table peuvent être associées à une clé unique.



26.2 Organisation physique

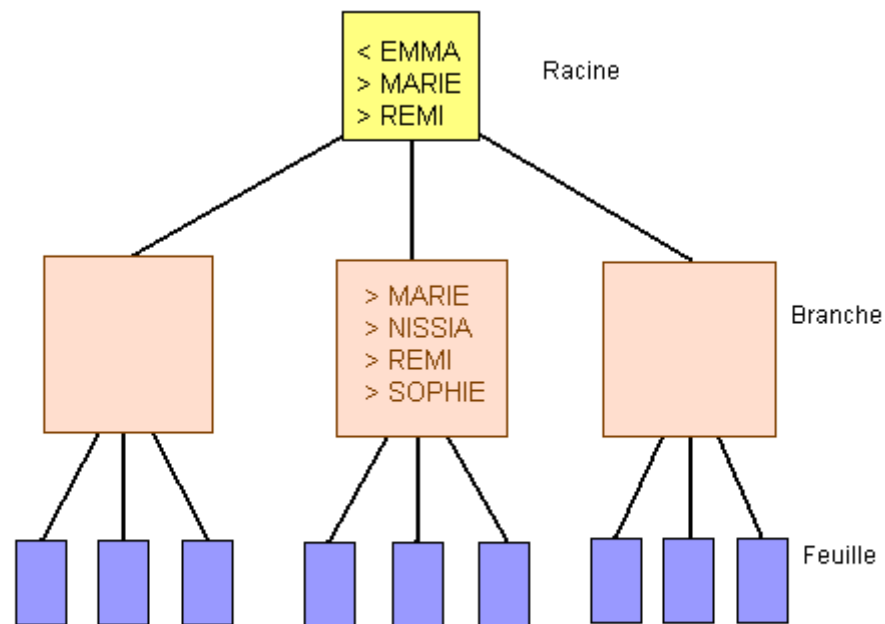
La structure de données utilisée par Oracle pour stocker un index est un *B*-Tree* (arbre B).

Oracle gère physiquement, différemment les index suivants :

- ⇒ Les index B*-Tree
- ⇒ Les index à clé inversée
- ⇒ Les index Bitmap

Structure d'un index B*_Tree

- ◆ Le nœud principal d'un index est appelé nœud racine (*root node*).
- ◆ Le deuxième niveau des nœuds est constitué par les branches (*branch*).
- ◆ Le niveau le plus faible est constitué des feuilles (*leaf*).
Les feuilles sont liées entre elles par une liste doublement chaînée, permettant un parcours des feuilles dans les 2 directions.

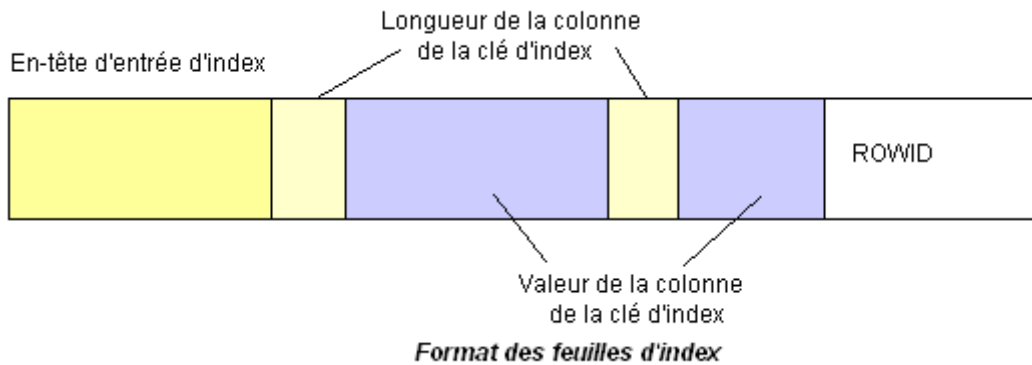


Index organisés en B-Tree*

En termes algorithmiques, la recherche dans un B*-Tree est semblable à celle réalisée dans un arbre binaire, à la différence qu'un arbre B*-Tree peut contenir jusqu'à *n* nœuds enfants, alors qu'un nœud d'un arbre binaire ne peut en contenir que 2.

Oracle n'utilise pas d'index organisés en nœuds binaires mais plutôt une arborescence équilibrée.





Les entrées d'index présentées ci-dessus ne concernent que les index globaux ou les tables non partitionnées.

Une entrée de feuille d'index se compose de :

- ♦ Un en-tête d'entrée qui stocke le nombre de colonnes et les informations sur le verrouillage.
- ♦ Des éléments de contrôle pour stocker la longueur de la colonne d'index.
- ♦ Les valeurs de la colonne d'index.
- ♦ Le ROWID de la ligne qui contient les valeurs de la clé d'index.

Dans un index organisé en *B*-Tree*, les valeurs de la clé sont répétées si plusieurs lignes de la table ont la même valeur de clé.

Les valeurs NULL ne figurent pas dans les entrées d'index.

Le ROWID est réduit car toutes les lignes appartiennent au même segment.

Lorsqu'une instruction SQL utilise un index, le nœud racine détermine le coté de l'arbre contenant la valeur recherchée. Les deux intermédiaires fournissent des informations de localisation des valeurs stockées dans chaque nœud de l'index.



Les index *B*-Tree* sont performants dans des environnements transactionnels (OLTP) pour l'indexation de tables dynamiques.

L'objectif d'un index est de réduire les entrées-sorties.

Cependant il arrive qu'un index provoque un nombre d'entrées-sorties supérieur au balayage complet d'une table.

Supposons une table contenant 1 million de lignes stockées dans 5 milles blocs, et que les lignes contenant une valeur donnée soient réparties sur plus de 4 milles blocs.

⇒ Dans ce cas il est préférable d'effectuer un balayage complet de table.



Même si le pourcentage brut de lignes renvoyées par la table est inférieur à 1%, dès lors qu'il faut parcourir 80% du nombre total de blocs de la table pour renvoyer les données, il est loin d'être optimal de créer et d'utiliser un index.

Si on additionne en plus, le nombre de blocs qu'il faut lire pour consulter l'index et extraire le ROWID, le coût d'utilisation de l'index augmente de manière vertigineuse.



L'utilisation d'un index ne doit pas être déterminée par un pourcentage arbitraire du nombre de lignes traitées ou sélectionnées dans une table, mais par le nombre de blocs qui doivent être lus pour renvoyer les données.

Si le nombre de blocs qui doivent être consultés pour un index est plus faible que celui d'un balayage complet de la table, l'index sera utile.

Chaque application et chaque base de données possèdent ses propres particularités, il faut donc éviter toute généralisation de sélectivité des lignes et de pertinence des index.

26.3 Accès par index B*-Tree

Rappels sur les règles d'utilisation d'un index B*-Tree :

- ⇒ Indexer les colonnes fréquemment utilisées dans les clauses `WHERE`
- ⇒ S'assurer que les requêtes utilisant la clé d'index sont sélectives : moins de 5 à 15% des lignes de la table extraites (dépend de la répartition des données dans la table)
- ⇒ Privilégier les index concaténés (□□attention à l'ordre des colonnes)
- ⇒ Ne pas hésiter à ajouter dans la clé d'index une colonne ramenée dans le `SELECT` (plus d'accès à la table !)
- ⇒ Indexer les clés étrangères (□□évite des problèmes de verrouillage sur la table enfant lors d'un `UPDATE` ou un `DELETE` sur la table père)
- ⇒ Ne pas indexer les petites tables
- ⇒ Gérer les index uniques à l'aide des contraintes `PRIMARY KEY` ou `UNIQUE`
- ⇒ S'assurer que l'écriture des requêtes n'empêche pas l'index d'être utilisé
- ⇒ S'assurer que les index créés ne dégradent pas les performances des mises à jour

Les colonnes fréquemment utilisées dans les clauses `WHERE` peuvent l'être comme critère de sélection ou critère de jointure.

En général, une sélectivité inférieure à 5% est bonne et une sélectivité supérieure à 15% est mauvaise ; entre les deux, il faut tester ...

Pour la sélectivité, il faut que les valeurs de la colonne soient relativement uniques (beaucoup de valeurs distinctes) et que les conditions qui les utilisent soient elles-mêmes sélectives.

Parmi les colonnes candidates, il faut d'abord identifier les colonnes qui sont systématiquement présentes ensemble dans la clause `WHERE` : ce sont de bonnes candidates pour la création d'un index composé qui est généralement plus sélectif qu'un index simple.



L'ordre des colonnes est important dans un index composé : un index composé est utilisé si les colonnes de tête de la clé d'index sont présentes dans la condition (mais l'ordre des colonnes dans la condition n'a pas d'importance).

Indexer les petites tables ne sert à rien car le nombre minimum de blocs à lire lors d'un accès par index est de 2 (1 bloc au minimum pour l'index et 1 bloc au minimum pour la table).

Grâce au paramètre `DB_FILE_MULTIBLOCK_READ_COUNT`, Oracle peut lire :

⇒ `DB_FILE_MULTIBLOCK_READ_COUNT` blocs en une entrée/sortie.

Donc, si la taille de la table est inférieure à `DB_FILE_MULTIBLOCK_READ_COUNT` blocs, un index est moins performant que le parcours complet.

Ainsi, en général, indexer des tables d'une vingtaine de blocs n'apporte rien.

Hormis les index uniques, il n'est jamais certain qu'un index soit réellement performant ; il faut donc tester !

Durant ces tests, il faut s'assurer que les index créés ne dégradent pas les performances des mises à jour.

26.4 Index sur fonctions

Index *B*-Tree* créé sur le résultat d'une expression. Il est apparu en version 8i.

Permet un traitement performant des ordres SQL qui utilisent des fonctions dans la clause `WHERE`.

Exemple

- Pour améliorer les performances de requêtes du type
`SELECT * FROM employe WHERE UPPER(nom) = 'CLO';`
- Créer l'index suivant
`CREATE INDEX indx_employe ON employe(UPPER(nom));`

Pré-requis :

- ⇒ Le paramètre `QUERY_REWRITE_ENABLED` doit être à `TRUE` (au niveau de l'instance ou de la session)
- ⇒ Les statistiques doivent être générées sur la table et sur l'index
- ⇒ L'optimiseur doit être en mode `CBO` (mode par défaut)

Les mêmes règles d'utilisation qu'un index B-Tree (sélectivité notamment) peuvent être appliquées.

L'idée de l'index sur une fonction, est d'indexer non pas la valeur de la colonne mais le résultat d'une fonction (plus généralement une expression) appliquée à la colonne.

Ce type d'index peut être créé sur une fonction applicative développée en PL/SQL.



26.5 Index Bitmap

Un index Bitmap est également organisé en *B*-Tree*, mais chaque nœud de la feuille stocke un *Bitmap* pour chaque valeur de la clé au lieu d'une liste de ROWID.

Chaque BIT du Bitmap correspond à un ROWID éventuel :

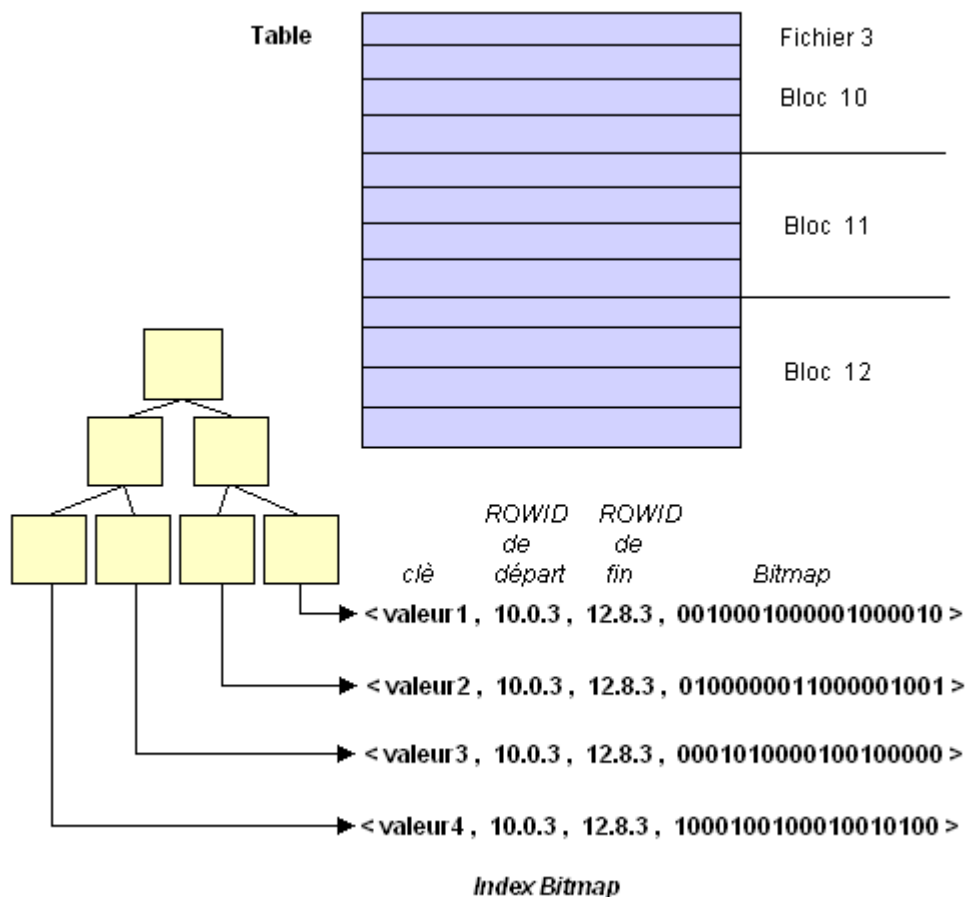
⇒ si le BIT est défini, alors la ligne comportant le ROWID correspondant contient la valeur de la clé.

Les index *Bitmap* utilisent des ROWID réduits.

Chaque BIT du *Bitmap* correspond à un ROWID éventuel :

⇒ Si le BIT est défini, alors la ligne comportant le ROWID correspondant contient la valeur de la clé.

L'index organisé en *B*-Tree* est utilisé pour localiser les nœuds de feuilles contenant des segments de *Bitmap* pour une valeur donnée de la clé. Le ROWID de départ et les segments de *Bitmap* sont utilisés pour localiser les lignes contenant la valeur de la clé.



Le nœud de la feuille d'un index *Bitmap* contient :

- ◆ Un en-tête d'entrée qui contient le nombre de colonnes et les informations sur le verrouillage.
- ◆ Les valeurs de la clé qui se compose de la longueur et des couples de valeur pour chaque colonne.
- ◆ Le ROWID de départ, contenant un numéro de fichier, un numéro de bloc, et un numéro de ligne.
C'est celui de la première ligne pointée par le segment de *Bitmap*, c'est à dire que le premier *Bitmap* correspond à ce ROWID, le deuxième correspond à la ligne suivante dans le bloc et le ROWID de fin pointe vers la dernière ligne de la table couverte par le segment de *Bitmap*.
- ◆ Le ROWID de fin contenant un numéro de bloc et un numéro de ligne
- ◆ Un segment de *Bitmap* composé d'une chaîne de BIT.
Un BIT est défini lorsque la ligne correspondante contient la valeur de la clé (il ne l'est pas lorsque la ligne ne contient pas la valeur de la clé). Le serveur Oracle utilise une technique de compression brevetée pour stocker les segments de *Bitmap*.

Lorsque des modifications sont apportées à la colonne de la clé dans la table, les *Bitmaps* doivent être modifiés.

Oracle effectue un **verrouillage au niveau segment de *Bitmap***, contrairement aux index *B*-Tree* pour lesquels le verrouillage ne s'applique qu'aux entrées correspondant aux lignes individuelles de la table.

Ainsi, une ligne couverte par le *Bitmap* ne peut pas être mise à jour par d'autres transactions pendant la durée de verrouillage.

Les index *Bitmap* sont performants pour :

- ⇒ Les tables comportant des millions de lignes et des colonnes de clé à faible cardinalité (par exemple le sexe ou l'état civil).
- ⇒ Les requêtes qui utilisent des prédicats OR (Oracle utilise le segment de *Bitmap* pour effectuer le OR).
- ⇒ Les environnements décisionnels où les requêtes sont complexes et effectuées sur de grandes tables statiques.

Utilisation :

Plutôt en décisionnel.

Indexer les colonnes à faible cardinalité plutôt utilisées dans des conditions multiples combinées par des AND et des OR.



A supprimer et à recréer lors des mises à jour massives de la table.

Le cas typique d'utilisation est le système décisionnel dans lequel une grosse table de « faits » (historique des ventes par exemple) est jointe en « étoile » avec beaucoup de petites tables de « dimension » utilisées comme axes d'analyse (produit, modèle, région, couleur, gamme, ...) ; généralement, les tables de dimension ont un relativement petit nombre de valeurs distinctes. Dans ce genre de système, la table est très fréquemment accédée avec des clauses combinant (AND et OR) plusieurs conditions sur les tables de dimension.



26.6 Index IOT

Toute la table est stockée dans l'index *B*-Tree* de la clé primaire.

L'accès s'effectue donc comme dans un index *B*-Tree* mais Oracle n'a plus besoin d'accéder à une deuxième structure.

Caractéristiques d'un index *IOT* :

- ⇒ Pas de *ROWID*
- ⇒ A partir de la 8i, possibilité d'avoir des index secondaires et des contraintes uniques sur les autres colonnes de la table (mais moins performants)
- ⇒ Gain en espace



Performant pour les accès par égalité ou intervalle sur la clé primaire.

Oracle n'a pas besoin d'accéder à la table si toutes les colonnes manipulées par une requête sont présentes dans la clé de l'index utilisé.

L'IOT repose sur le même principe puisqu'il consiste à stocker la table dans les blocs de l'index utilisé pour la clé primaire ; toutes les colonnes étant dans cet index, Oracle n'a plus de deuxième structure à lire après l'index.

Bien qu'il soit possible de créer des index secondaires sur un *IOT*, les *IOT* sont surtout intéressants pour les tables auxquelles on accède principalement par l'intermédiaire de la clé primaire (ou toute clé constituant un préfixe de la clé primaire dans le cas d'une clé concaténée).

Ce type d'index est utilisé pour :

- ⇒ Des tables auxquelles on accède uniquement par la clé primaire
- ⇒ Des tables dont la taille de la ligne est inférieure à la moitié d'un bloc

Il convient de réserver ce type de structure à des tables dont la longueur des lignes est plutôt petite (du type table de nomenclature, comportant un code et un libellé).

Pour pouvoir utiliser des *IOT* pour des tables ayant de grandes lignes, Oracle propose :

- ♦ la clause *OVERFLOW* qui permet de spécifier un stockage pour un segment de débordement
- ♦ la clause *PCTTHRESHOLD* qui permet de spécifier une limite exprimée en pourcentage de la taille du bloc

Si la taille de la ligne est supérieure à la limite ($PCTTHRESHOLD \times DB_BLOCK_SIZE / 100$) alors les colonnes qui ne font pas partie de la clé primaire sont stockées en dehors de l'index, dans un segment de débordement séparé défini par la clause *OVERFLOW*.

Cette technique permet de conserver un parcours d'index performant mais une deuxième structure physique doit être lue si les colonnes manipulées dans la requête ne sont pas toutes dans la clé primaire ; le niveau de performance de l'utilisation de l'*IOT* dépend de la nature des requêtes et du nombre de lignes utilisant de l'espace dans le segment de débordement.



```
CREATE TABLE nom_table  
(  
  spécification colonnes,  
  CONSTRAINT nom_contrainte PRIMARY KEY (colonnes clé primaire)  
)  
ORGANIZATION INDEX  
[ TABLESPACE tablespace_index_clause STORAGE]  
[ PCTTHRESHOLD valeur ]  
[ INCLUDING nom_colonne ]  
[ OVERFLOW TABLESPACE tablespace_débordement_clause_STORAGE]  
;
```

Un IOT se crée par un `CREATE TABLE` suivi de l'option `ORGANIZATION INDEX` ; le `CREATE TABLE` doit obligatoirement contenir la définition de la clé primaire.

Depuis la version 8i, Oracle permet de créer des index secondaires (de type B*-Tree) sur un IOT ... malgré le fait qu'une ligne stockée dans un IOT n'a pas de ROWID.

Pour contourner cette absence, Oracle utilise un ROWID logique qui permet de « deviner » où la ligne est physiquement stockée.

La performance d'un index secondaire aura tendance à se dégrader au fur et à mesure que les données de l'IOT sont mises à jour et que les lignes concernées changent de bloc ; il convient donc de reconstruire les index secondaires régulièrement pour maintenir un bon niveau de performance.

26.7 Index à clé inversée

Un index à clé inversée inverse les octets de chaque colonne indexée (sauf le ROWID) tout en conservant l'ordre de la colonne.

Il est apparu en version 8.



Les index à clé inversée servent seulement pour les requêtes contenant des prédicats d'égalité.

Les index à clé inversée répartissent la distribution des mises à jour d'index sur l'ensemble de l'arborescence de l'index en inversant la valeur des clés d'index.

Lors de l'insertion de lignes sur une clé ascendante gérée par un index *B*-Tree*, des goulots d'étranglement peuvent se produire sur l'index car toutes les mises à jour d'index s'effectuent au même emplacement dans l'arborescence de l'index.

La valeur de la clé d'index 8967 est stockée dans l'index pour l'identifiant 7698.
A la saisie du numéro d'identifiant suivant, 7782, une entrée d'index est créée pour 2877.

La charge de travail d'Oracle est répartie sur plusieurs blocs d'index.



Les recherches d'intervalles ne peuvent pas être exécutées à l'aide d'un index à clé inversée car les clés adjacentes ne sont pas stockées l'une à côté de l'autre.



Index sur table Article (Id_code)

Table Article

Index sur table Article (Id_code)				Table Article		
CLE	ROWID			Id_code	Nom	Prix
Id_code	N°Bloc	N°Ligne	N°Fichier			
1257	0000000F	0002	0001	7499	crayon noir	02,10
2877	0000000F	0006	0001	7369	crayon rouge	02,20
6567	0000000F	0004	0001	7521	crayon bleu	02,10
6657	0000000F	0003	0001	7566	crayon jaune	02,22
8967	0000000F	0005	0001	7698	crayon vert	02,15
9637	0000000F	0001	0001	7782	crayon rose	02,30
---				---		
---				---		
---				---		

Index à Clé inversée

Accès par index à clé inversée

Index *B*-Tree* qui inverse les octets de chaque colonne de la clé.

L'index est parcouru comme un *B*-Tree* classique.

Caractéristiques :

- ⇒ 2 clés consécutives ne sont plus contiguës dans les blocs d'index

Performant pour :

- ◆ Les insertions concurrentes massives car moins de contention sur les blocs d'index
- ◆ Pas d'avantage particulier sur les recherches

Cet index est utilisé pour les clés à valeur croissante et à insertions concurrentes massives utilisées uniquement en recherche d'égalité.

```
CREATE INDEX nom ON (liste_colonnes) REVERSE  
;
```

En inversant les octets de la clé d'index, les insertions de valeurs de clé consécutives ne sont plus concentrées dans les mêmes blocs feuilles mais réparties dans plusieurs blocs feuilles de l'index.

Si les insertions concernées sont massivement concurrentes, cette technique permet d'éviter les contentions sur les blocs d'index.

Par contre, deux valeurs de clé consécutives ne sont plus stockées côte à côte, ce qui ne permet plus d'utiliser l'index pour des recherches par plage de valeur (range scan).



26.8 Créer et Spécifier le stockage d'un index

Le stockage d'un index peut être spécifié lors de la création de l'index (`CREATE INDEX`) ou de la contrainte de clé primaire ou unique (clause `CONSTRAINT`).

```
CREATE [UNIQUE] INDEX nom_index ON nom_table(liste des colonnes)
[ TABLESPACE nom_tablespace ]
[ PARALLEL degrés ]
[ PCTFREE valeur ]
[ clause_stockage ]
;
```

♦ clause CONSTRAINT

```
CONSTRAINT nom_contrainte ...
USING INDEX
[ TABLESPACE nom_tablespace ]
[ PARALLEL degrés ]
[ PCTFREE valeur ]
[ clause_stockage ]
```

♦ clause_stockage

```
STORAGE ( [ INITIAL valeur [K|M] ]
[ NEXT valeur [K|M] ]
[ MINEXTENTS valeur ]
[ MAXEXTENTS { valeur | UNLIMITED } ]
[ PCTINCREASE valeur ] )
```

- **UNIQUE** = indique que l'index est un index unique (non unique par défaut)
- **TABLESPACE** = nom du tablespace dans lequel l'index doit être créé
- **PARALLEL** = degrés de parallélisme
- **PCTFREE** = valeur du **PCTFREE** (entre 0 et 100, 10 par défaut)
- **STORAGE** = clause de stockage de l'index
- **INITIAL** = taille du premier extent alloué
- **NEXT** = taille du deuxième extent alloué
- **MINEXTENTS** = nombre initial d'extent(s) alloué(s)
- **MAXEXTENTS** = nombre maximal d'extents allouables
- **PCTINCREASE** = pourcentage d'augmentation (0 à 100) de la taille des extents, à partir du troisième, par rapport au précédent

```
• Lors de la création de l'index
CREATE INDEX employe_nom_prenom ON employe(nom,emploi)
TABLESPACE indx
PCTFREE 20
STORAGE ( INITIAL 2000K NEXT 400K MAXEXTENTS 64 PCTINCREASE 0 ) ;
```



```
• Lors de la création d'une contrainte
-- =====
• Table : EMPLOYE
-- =====
create table EMPLOYE
(
    ID_EMP      INTEGER          not null,
    NOM         VARCHAR2(30)     not null,
    SALAIRE     NUMBER(4)        not null,
    EMPLOI      VARCHAR2(20)     null ,
    EMP_ID_EMP  INTEGER          null ,
    constraint PK_EMPLOYE primary key (ID_EMP)
    using index
    tablespace INDX
)
tablespace DATA
/

-- =====
• Index : CLES ETRANGERES
-- =====
alter table EMPLOYE
add constraint FK_EMPLOYE_A_POUR_PA_EMPLOYE foreign key (EMP_ID_EMP)
references EMPLOYE (ID_EMP)
USING INDEX
TABLESPACE indx
PCTFREE 20
STORAGE ( INITIAL 2000K NEXT 400K MAXEXTENTS 64 PCTINCREASE 0 )
/
```



Avoir un grand nombre d'extents ne joue pas sur les performances.
Si la clause « tablespace » n'est pas spécifiée, c'est le tablespace de
l'utilisateur qui sera pris par défaut (voire le tablespace system).

26.8.1 La clause USING INDEX

Elle permet de mentionner explicitement le nom d'un index existant à utiliser pour vérifier la contrainte :

```
| USING INDEX nom_index
```

Elle permet d'inclure un ordre SQL CREATE INDEX pour créer explicitement l'index associé à la contrainte :

```
| USING INDEX (création_index)
```

- Où **création_index** est un ordre SQL CREATE [UNIQUE] INDEX classique.

```
SQL> alter table employe
2      add CONSTRAINT employe_nom_UNIQUE UNIQUE (nom)
3      USING INDEX
4      (
5          CREATE INDEX employe_nom ON employe(nom)
6              TABLESPACE indx
7              STORAGE (INITIAL 10M NEXT 10M PCTINCREASE 0)
8      )
9      ;

Table modifiée.
SQL> insert into employe values
2      (100, 'Martin',1000,'Prof',null);
```



```
1 ligne cr  e.

SQL> insert into employe values
2 (101, 'Martin',1000,'Prof',null);
insert into employe values
*
ERREUR   la ligne 1 :
ORA-00001: violation de contrainte unique (OPDEF.EMPLOYE_NOM_UNIQUE)
```

L'index mentionn  ou cr   peut  tre unique ou non unique.

L'index mentionn  ou cr   doit  tre « compatible » avec la contrainte de cl  primaire ou unique :

- ⇒ Si l'index est unique, la cl  de l'index doit  tre  gale   la cl  de la contrainte (m mes colonnes, dans le m me ordre)
- ⇒ Si l'index est non unique, la cl  de l'index doit  tre  gale   ou commencer par la cl  de la contrainte



La cl  de l'index associ  ne peut pas  tre plus « courte » (i.e. avec moins de colonnes) que la cl  de la contrainte, car dans ce cas, il n'y a pas assez d'informations dans l'index pour v rifier la contrainte.

Dans le cas de la cr ation de l'index, une erreur est retourn e s'il existe d j  un index sur la cl .

Fonctionnellement, cr er l'index avant la contrainte et le mentionner dans l'ordre de cr ation de la contrainte est strictement  quivalent   cr er l'index dans l'ordre de cr ation de la contrainte.

Estimer la taille de chaque index et sp cifier la clause `STORAGE`.

- ⇒ Essayer de limiter le nombre d'extents de chaque index
- ⇒ Pr f rer un petit nombre de gros extents (voire un seul gros extent) plut t qu'un grand nombre de petits extents

Taille des extents :

- ⇒ Pr f rer des tailles multiples de `5 x DB_BLOCK_SIZE`

Positionner `PCTFREE` avec soin.

Ne pas oublier que le premier extent fait toujours au minimum 2 blocs (pr sence d'un bloc d'en-t te).

Allouer un extent `INITIAL` d'une taille adapt e   la volum trie estim e   une  ch ance donn e (1 an, 2 ans, etc ...).



Surveiller les premiers mois d'exploitation et rectifier la valeur de NEXT en cas de besoin.

```
SQL> col index_name format A20
SQL> col table_name format A14
SQL> col tablespace_name format A5
SQL> col initial_extent format 999999999

SQL> select INDEX_NAME, TABLE_NAME, TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT
2   from dba_indexes
3  where owner='OPDEF';
```

INDEX_NAME	TABLE_NAME	TABLESPACE_NAME	INITIAL_EXTENT	NEXT_EXTENT
A_POUR_PATRON_FK	EMPLOYE	INDX	65536	
EMPLOYE_NOM	EMPLOYE	INDX	2048000	
EQUIPAGE_FK	EST_EQUIPAGE	INDX	65536	
EST_EQUIPAGE_FK	EST_EQUIPAGE	INDX	65536	
PK_AVION	AVION	INDX	65536	
PK_EMPLOYE	EMPLOYE	INDX	65536	
PK_EST_EQUIPAGE	EST_EQUIPAGE	INDX	65536	
PK_VOL	VOL	INDX	65536	
UTILISE_FK	VOL	INDX	65536	

26.9 Calcul de PCTFREE

Ne pas se préoccuper de PCTFREE

- ⇒ Si la colonne indexée est vide lors de la création de l'index

Positionner PCTFREE à une valeur faible (voir égale à 0)

- ⇒ Si l'index est créé sur une colonne qui sera rarement mise à jour (ni UPDATE ni INSERT)
- ⇒ Si l'index est créé sur une colonne qui va continuer à faire l'objet d'insertions avec des valeurs NULL (ne vont pas dans l'index) ou supérieures aux valeurs existantes (vont dans des nouveaux blocs)

Positionner PCTFREE à une valeur élevée

- ⇒ Si l'index est créé sur une colonne qui sera mise à jour (UPDATE) ou qui va continuer à faire l'objet d'insertions avec des valeurs appartenant à la plage des valeurs actuelles
- ⇒ $PCTFREE = 100 \times (1 - N_i / N_f)$
 - N_i = nombre initial de lignes
 - N_f = nombre final de lignes (à une échéance donnée)



Pour mémoire, PCTFREE est pris en compte uniquement à la création de l'index et ne sera effectivement utilisé que si la colonne à indexer est non vide.



26.10 Rappel sur l'analyse des index

Le seul élément vraiment significatif est le nombre de *blocs de feuilles* qui doivent être lus. Plus ce nombre est faible plus le nombre d'entrées-sorties le sera et plus grande sera la vitesse de lecture des lignes de tables.

➡ Plus il y aura d'insertions ou de suppressions dans une table et plus le risque de fragmentation sera élevé.

DBMS_STATS.AUTO_SAMPLE_SIZE permet de maximiser les gains d'exécution tout en réalisant avec exactitude les statistiques nécessaires.

Cependant, si les index sont analysés en même temps que les tables, le niveau de parallélisme qui peut s'appliquer au calcul des statistiques sur les tables ne s'applique pas aux index.

Si ceux-ci doivent être analysés en parallèle, il vaut mieux exécuter indépendamment la commande GATHER_INDEX_STATS.

```
Exec dbms.stats.gather_schema_stats (
  'CHARLY',DBMS_STATS.AUTO_SAMPLE_SIZE
);

Execute dbms_stats.gather_schema_statistics('clo01',
20,estimate_percent=>20);
```

Dans l'analyse finale, vérifiez que les statistiques ont été calculées pour tous les index afin d'éviter des statistiques incomplètes dans la base de données.



Il faut savoir qu'un bloc d'index vide n'est pas réutilisé tant que l'index n'est pas compacté ou reconstruit.

Les statistiques générées sur les index alimentent les colonnes de la table DBA_INDEXES

DBA_INDEXES

- ✓ **BLEVEL**, = Profondeur de l'arbre au niveau des branches (ne tient pas compte des feuilles). 0 si le bloc racine est égal au bloc feuille. Valeur exacte même en ESTIMATE
- ✓ **LEAF_BLOCKS** = Nombre de blocs feuilles dans l'index
- ✓ **NUM_ROWS** = Nombre de lignes dans l'index
- ✓ **DISTINCT_KEY** = Nombre de valeurs distinctes dans l'index
- ✓ **SAMPLE_SIZE** = Taille de l'échantillon utilisé en cas d'analyse ESTIMATE
- ✓ **LAST_ANALYZED** = Date et heure de la dernière analyse réalisée sur l'index



```
SQL> select INDEX_NAME, BLEVEL, LEAF_BLOCKS, NUM_ROWS
2  from dba_indexes
3  where index_name='PK_AVION'
4  ;
```

INDEX_NAME	BLEVEL	LEAF_BLOCKS	NUM_ROWS
PK_AVION	0	1	3



La hauteur d'un index est un élément clé de réduction du nombre d'entrées-sorties générées par cet index.

La génération de statistiques permet de donner suffisamment d'informations à l'optimiseur CBO sur les index.

26.10.1 Problèmes détectés

Deux problèmes peuvent être détectés :

- ⇒ Faible taux d'occupation et/ou profondeur importante de l'index

Profondeur importante de l'index

BLEVEL est élevé (supérieur à 5).

- ⇒ Lié à un PCTFREE mal adapté lors de la création ou à un index très volatile (beaucoup de mises à jour)
- ⇒ Dégrade les performances de l'utilisation de l'index.

26.11 Réorganiser le stockage d'un index

Un index peut être réorganisé pour :

- ⇒ Libérer de l'espace libre au dessus de la HWM
- ⇒ Réorganiser un index dont la structure s'est dégradée
- ⇒ Réorganiser plus globalement le stockage de l'index (changement de tablespace, modification de la clause STORAGE, reconstruction avec un nouveau paramètre PCTFREE, réduction du nombre d'extents alloués, ...)

Typiquement lorsqu'une application met un temps beaucoup plus important qu'à son habitude et ne voit pas la fin de son exécution, il faut réorganiser les index car il y a baisse des performances dues à la dégradation des index.

La réorganisation avant la relance de l'application permettra un gain de performance énorme.



Il y a différentes techniques :

- ◆ Ordre SQL `ALTER INDEX ... DEALLOCATE UNUSED`, à utiliser pour libérer de l'espace libre au dessus de la HWM
- ◆ Recréer l'index : ordres SQL `DROP INDEX` puis `CREATE INDEX`, à utiliser pour les réorganisations
- ◆ Reconstruire l'index : ordre SQL `ALTER INDEX ... REBUILD`, à utiliser pour les réorganisations
- ◆ Reconstruire l'index : ordre SQL `ALTER INDEX ... COALESCE`, à utiliser pour fusionner des segments d'index
- ◆ Réorganiser l'index : ordre SQL `ALTER INDEX ... SHRINK SPACE`, à utiliser pour les réorganisations de segments

Réorganiser le stockage d'un index est moins compliqué que de réorganiser le stockage d'une table car les données ne sont pas impactées et la table est toujours accessible et opérationnelle.



Lors d'un traitement massif sur une table (chargement, purge), il peut être intéressant de supprimer tout ou partie des index de la table avant le traitement et de les recréer après. La performance sera meilleure et l'index sera tout neuf et non dégradé.

26.11.1 Comment reconstruire des index dégradés

La dégradation des index se manifeste par la lenteur d'exécution des applications, qui deviennent de plus en plus lentes au fur et à mesure que les index se dégradent. Les performances sont immédiatement améliorées après reconstruction des index.

Il faut planifier soigneusement la reconstruction d'un index. Lors de cette opération, il faut être rapide et efficace.

Avant la version 8i d'Oracle il fallait un temps d'arrêt. Depuis la version 8i, il est possible de le faire en ligne.

Les éléments qui peuvent vous aider pendant la reconstruction d'index sont :

- ⇒ Le parallélisme
- ⇒ L'annulation de génération de Redo Log

Le premier est obtenu par le positionnement de la clause `PARALLEL` dans la commande `ALTER INDEX ... REBUILD`.

Le second par l'utilisation de la clause `NOLOGGING` (à partir de la version 8.0 d'Oracle).

```
Alter index equipage_FK rebuild
Parallel (degree 4)
Nologging
Tablespace INDX
;
```





Prévoir une sauvegarde du tablespace contenant les index après reconstruction de ceux-ci, en cas de défaut de média, sur ce tablespace avant la prochaine sauvegarde.

Il est judicieux de compacter les nœuds des feuilles d'un index en utilisant l'option `COALESCE` dans l'instruction `ALTER INDEX`, afin que les blocs libres puissent être réutilisés.

Utilisez cette option sur les index qui subissent un volume significatif d'opérations d'insertion (`INSERT`) et de suppression (`DELETE`).

26.11.2 L'ordre SQL `ALTER INDEX ... REBUILD`

L'ordre SQL `ALTER INDEX ... REBUILD` permet de reconstruire un index et si besoin de réorganiser son stockage :

```
ALTER INDEX nom index REBUILD  
[ TABLESPACE nom tablespace ]  
[ PCTFREE valeur ]  
[ clause stockage ]  
;
```

♦ clause_stockage

```
STORAGE ( [ INITIAL valeur [K|M] ]  
          [ NEXT valeur [K|M] ]  
          [ MINEXTENTS valeur ]  
          [ MAXEXTENTS { valeur | UNLIMITED } ]  
          [ PCTINCREASE valeur ] )
```

Sans option, il reconstruit l'index avec les mêmes clauses de stockage.

```
ALTER INDEX employe_nom REBUILD  
PCTFREE 40  
STORAGE ( INITIAL 8000K NEXT 400K MAXEXTENTS 64 PCTINCREASE 0 )  
;
```

26.11.3 L'ordre SQL `ALTER INDEX ... COALESCE`

Cette commande permet de fusionner le contenu de blocs feuilles adjacents possédant de l'espace libre. Ainsi deux blocs feuilles adjacents possédant 50% d'espace libre peuvent fusionner en 1 bloc plein et 1 bloc libre.

Cette commande n'effectue aucune réorganisation de segment, il n'y a aucune amélioration sur la profondeur des branches. Par contre, dans de nombreuses situations cette commande est suffisante pour retrouver un index performant (il n'y a plus de trous).

```
ALTER INDEX nom index COALESCE  
;
```



26.11.4 L'ordre SQL ALTER INDEX ... SHRINK SPACE

Cette commande est identique à celle utilisée pour la réorganisation de segments de tables et suit les mêmes règles.

Elle permet de compacter les index stockés dans des tablespaces gérés localement avec une gestion des segments automatique.

```
ALTER INDEX NO_VOL_PK SHRINK SPACE ;
```

26.12 Surveiller l'utilisation des index

Oracle9i permet de surveiller les index afin de déterminer s'ils sont utilisés ou non :

Si un index s'avère non utilisé, il est peut être opportun de le supprimer.

En version Oracle 10g, les statistiques sont générées automatiquement et le monitoring d'un index est positionné par défaut, il permet de savoir s'il est utilisé ou non.

L'ordre SQL ALTER INDEX permet d'activer ou de désactiver la surveillance d'un index.

```
ALTER INDEX index MONITORING USAGE | NOMONITORING USAGE ;
```

La vue V\$OBJECT_USAGE peut être interrogée pour déterminer si un index a été utilisé pendant qu'il était sous surveillance. Elle contient les colonnes :

- INDEX_NAME = nom de l'index
- TABLE_NAME = nom de la table
- MONITORING = indique si l'index est sous surveillance (YES | NO)
- USED = indique si l'index a été utilisé au moins une fois pendant sa surveillance (YES | NO)
- START_MONITORING = date/heure du début de la surveillance de l'index
- END_MONITORING = date/heure de la fin de la surveillance de l'index (vide si la surveillance est en cours)

La vue V\$OBJECT_USAGE est réinitialisée lors de l'activation de la surveillance d'un index.

Lors de la désactivation de la surveillance d'un index, la date/heure de fin de la surveillance est simplement enregistrée dans la colonne END_MONITORING et les informations sont conservées.

Les colonnes START_MONITORING et END_MONITORING donnent la date et l'heure sous forme de chaîne, selon le format MM/DD/YYYY HH24:MI:SS



La vue V\$OBJECT_USAGE doit être interrogée sous le compte du propriétaire de l'index (pas de colonne OWNER dans la vue).



La colonne `USED` de la vue `V$OBJECT_USAGE` indique simplement si l'index a été utilisé au moins une fois pendant la période de surveillance :

- ⇒ Combien de fois ?
- ⇒ La période de surveillance est-elle représentative ?
- ⇒ Si l'index est utilisé, est-il réellement intéressant ?

Une analyse complémentaire doit être menée pour décider de conserver ou de supprimer un index.

Si un index est déclaré comme non utilisé, il faut avant tout se poser la question de la représentativité de la période analysée. L'index est peut-être utilisé à d'autres moments, peut-être une seule fois par mois lors de la production d'un rapport, et il peut être très important à ce moment là. Le supprimer n'est donc peut-être pas une bonne idée.

Lorsque l'index est déclaré comme utilisé, il manque des informations sur la fréquence de son utilisation, sur sa pertinence réelle pour les performances, sur son impact négatif éventuel sur les mises à jour, etc.

Le conserver n'est donc peut-être pas une bonne idée.



26.13 Supprimer un index

Pour supprimer un index, on utilise la commande SQL `DROP INDEX`

```
DROP INDEX nom ind
;
```

Par défaut, lorsque qu'une clé primaire ou unique est supprimée, l'index associé l'est aussi s'il est unique, mais il n'est pas supprimé s'il est non unique.

Il est possible d'indiquer explicitement si l'index associé à une contrainte supprimée doit être supprimé ou non.

```
ALTER TABLE nom
DROP CONSTRAINT {nom contrainte | PRIMARY KEY}
KEEP INDEX | DROP INDEX
;
```

```
SQL> alter table employe
2 drop CONSTRAINT employe_nom_UNIQUE
3 drop index;
```

Table modifiée.



Conserver un index unique lors de la suppression d'une contrainte unique ou primary key n'a pas de sens car l'unicité est toujours vérifiée.



26.14 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur les index :

- DBA_INDEXES : informations sur les index
- DBA_IND_COLUMNS : informations sur les colonnes des index
- INDEX_STATS : résultat du dernier ANALYZE INDEX ... VALIDATE STRUCTURE
- DBA_SEGMENTS : informations sur les segments (dont ceux de type index)
- DBA_EXTENTS : informations sur les extents alloués à l'index

```
PROMPT--analyse de la structure des index du schéma TAHITI
• utilisation d'une table de même structure que index_stats
• destinée à stocker les statistiques de plusieurs index
DROP TABLE multi_index_stats
/
SQL> CREATE TABLE multi_index_stats
2 AS SELECT * FROM index_stats WHERE 0=1
3 /

Table créée.

-- premier programme PL/SQL (avec SQL dynamique) pour analyser
• la structure de tous les index du schéma TAHITI
SQL> DECLARE
2 CURSOR c_index IS
3 SELECT owner || '.' || index_name nom_index FROM dba_indexes
4 WHERE owner = 'OPDEF';
5 BEGIN
6 FOR l IN c_index LOOP
7 EXECUTE IMMEDIATE
8 'ANALYZE INDEX ' || l.nom_index || ' VALIDATE STRUCTURE';
9 INSERT INTO multi_index_stats
10 SELECT * FROM index_stats;
11 END LOOP;
12 END;
13 /
```



27 Les partitions de tables et d'index

La notion de partition permet de pouvoir répartir des données sur plusieurs segments d'un même objet.

Le partitionnement permet de gérer de grosses bases de données de plusieurs téra-octets, en découpant les grosses tables et index en morceaux plus petits.

A partir de la version 8 d'Oracle, il existe les tables partitionnées, qui permettent le développement d'applications évolutives car elles comprennent plusieurs partitions pouvant être situées dans des tablespaces différents.

Les partitions sont utiles pour de grandes tables qui peuvent être interrogées à l'aide de plusieurs processus simultanés.

Il existe également, des index partitionnés pour stocker les entrées d'index correspondant à un index dans plusieurs segments.

Les index partitionnés sont généralement utilisés pour les tables partitionnées. Une partition d'index peut être créée pour chaque partition de table



Le partitionnement est intéressant à condition de ne balayer qu'une partition dans une requête SQL.

Il existe 3 modes de partitionnement :

- ♦ Par intervalle de valeur (by range)
- ♦ Par liste de valeur (by list)
- ♦ Par fonction de hashage (by hash)

27.1.1 Performance des partitions

Les partitions peuvent être :

- ♦ Déplacées d'un tablespace vers un autre
- ♦ Supprimées, ajoutées ou divisées
- ♦ Individuellement déplacées, réorganisées ou rechargées

Si une partition est indisponible, les autres continuent à être disponibles, par exemple en cas de crash disque.

Il y a réduction du temps de restauration online en cas d'incident sur un tablespace, l'unité de restauration étant plus petite.



Par contre, la requête ne doit balayer qu'une partition à la fois.

```
Create table article
( no_vente    integer,
  quantite    integer,
  libelle     varchar2(50),
  famille     varchar2(100),
  date_vente  date,
  primary key no_vente
)
--- attributs physiques de la table -
storage (initial 10K next 20K)
tablespace tbs_ventes

partition by list (famille)
( partition linge values ('serviettes',
'torchons',
'draps')
storage (initial 20K next 40K pctincrease 50)
tablespace tbs_linge)

( partition frais values ('lait',
'fruits',
'legumes',
'viandes') )

( partition droguerie values ('detergents',
'peintures')
);
```

Il y a gain de performances :

- ⇒ Réduction des accès disque avec parallélisme des I/O
- ⇒ Elimination des accès à certaines partitions

```
Select * from articles where famille = 'fruits' ;
Select * from articles where famille = 'serviettes' ;
```

Règles de nommage :

Chaque partition doit avoir un nom unique.

Si aucun nom n'est indiqué, Oracle lui attribue automatiquement un nom au format SYS_Pn (n = numéro unique dans la base).

Valeurs des partitions :

Elles sont non inclusives.

Chacune exceptée la première a sa borne de valeur spécifiée par les partitions précédentes.

Une insertion de ligne échoue si la valeur spécifiée est plus haute que la valeur spécifiée pour la plus haute partition.



Cas particuliers :

Les colonnes de type LONG, LONG RAW et ROWID et les colonnes objets ne peuvent pas être utilisées comme clés de partitionnement.

Toutes les partitions d'une table partitionnée doivent résider dans des tablespaces possédant la même taille de bloc.

27.2 Vues du dictionnaire de données

Les vues suivantes permettent d'obtenir des informations sur les tables et index partitionnés :

- DBA_PART_TABLES = informations sur les tables partitionnées
- DBA_PART_INDEXES = informations sur les index partitionnés
- DNA_PART_KEY_COLUMNS = clés de partitionnement des tables partitionnées
- DBA_TAB_PARTITIONS = informations sur les partitions des tables partitionnées
- DBA_IND_PARTITIONS = informations sur les partitions des index partitionnés
- DBA_PART_COL_STATISTICS = statistiques sur les colonnes des tables partitionnées
- DBA_PART_HISTOGRAMS = histogrammes de partition des valeurs des tables partitionnées.



28 Le Scheduler (CJQ)

Job Queue Coordinator (CJQ), utilisé par le *Scheduler*, génère les processus pour exécuter les jobs planifiés qui se trouvent dans la file d'attente interne d'Oracle.

Les utilisateurs peuvent créer des jobs et les soumettre à ce coordinateur.



Le paramètre `JOB_QUEUE_PROCESSES` > 0 il permet de définir le nombre de job soumis en simultané.

Ce processus permet l'automatisation de tâches dans la base de données.

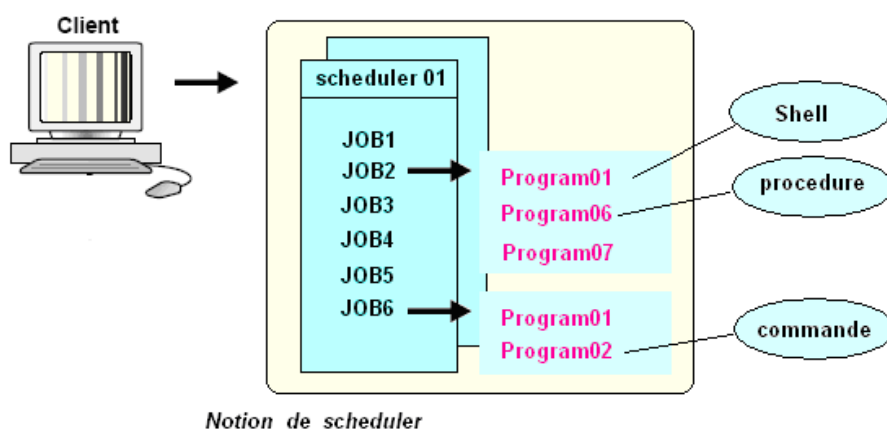
En effet, il est intéressant d'avoir la possibilité d'effectuer des travaux en différé, tels que :

- ♦ La génération de statistiques sur des tables et des index
- ♦ La réplication de données
- ♦ La gestion des sauvegardes
- ♦ Lister les événements en attente depuis un certain temps
- ♦ Exécuter un Batch
- ♦ ...

Aujourd'hui, des applications BtoB (*Business to Business*) demandent un suivi régulier et rigoureux. Il peut s'avérer nécessaire d'exécuter des *Batches* afin de répartir l'information.

A terme, il doit remplacer `DBMS_JOB` utilisé dans la réplication de données. Ainsi on pourra faire cohabiter des jobs anciens et nouveaux, ce qui n'est pas encore le cas aujourd'hui.

Le package `DBMS_SCHEDULER` permet d'avantage de possibilités que le package `DBMS_JOB` qui était une première version d'exécuteur de travaux.



Le Scheduler permet de spécifier :

- ♦ Un **JOB** définit ce qui doit être exécuté : le **QUOI** + **QUAND**
Par exemple, s'il s'agit d'un programme Java, shell ou pl/sql.
Il est possible de spécifier le programme et de le planifier comme faisant partie de la définition du **JOB** ou vous pouvez utiliser un **SCHEDULE** existant ou un **PROGRAM** existant.
Il est possible de passer des arguments à un **JOB** afin de personnaliser son comportement.
- ♦ Un **SCHEDULE** définit **QUAND** et **COMBIEN** de fois l'action doit être exécutée, il peut s'appliquer à plusieurs **JOB**.
Spécifie quand et combien de fois un **JOB** doit être exécuté.
Vous pouvez stocker le **SCHEDULE** pour un **JOB** séparément et ainsi utiliser le même *Schedule* pour plusieurs **JOB**.
- ♦ Un **PROGRAM** définit ce que le **JOB** exécute.
C'est une collection de métadonnées d'un exécutable, d'un script ou d'une procédure. Un **JOB** automatisé exécute certaines tâches. En utilisant un **PROGRAM**, cela vous permet de modifier la tâche du **JOB** (le quoi) sans modifier le **JOB** lui-même.
Vous pouvez définir les arguments pour un **PROGRAM**, autorisant les utilisateurs à modifier le comportement de la tâche.

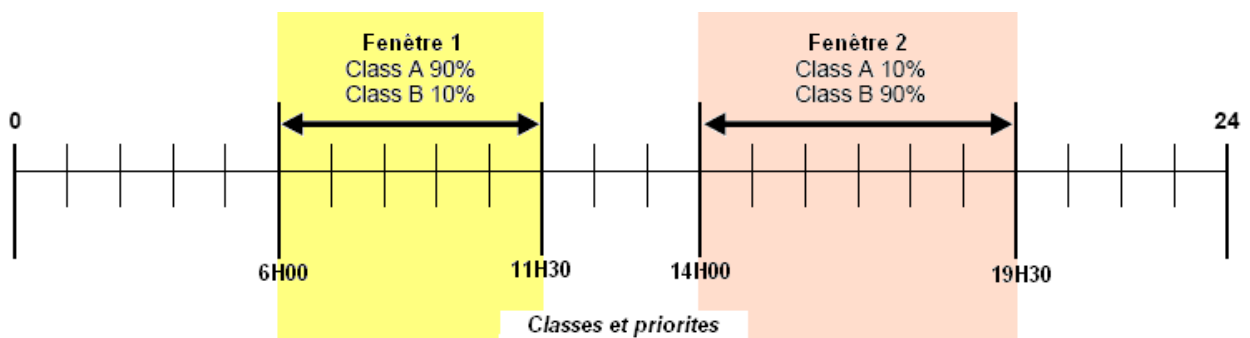


Chaque composant de *Scheduler* est un objet de schéma et doit avoir un nom unique.

Le *Scheduler* permet de définir des fenêtres ouvertes dans le temps, associées à des heures bien précises.

Dans ces fenêtres, il est possible de définir des *classes de priorités de Jobs*. A ces classes sont associés des pourcentages de consommation de :

- ⇒ Ressource machine
- ⇒ Plan d'exécution
- ⇒ Fenêtre de travail



Lorsque le *Job* est créé, il est placé dans une table du dictionnaire de données.

Le *Scheduler* utilise une table de *Jobs* par Database et un *Job Coordinateur* (JCQ) par instance.

Le JCQ est un processus d'arrière plan qui se réveille lorsque des *Jobs* doivent être exécutés, ou lorsqu'une fenêtre doit être ouverte. Ainsi lorsqu'un *Job* doit s'exécuter, le JCQ notifie automatiquement à un processus esclave d'effectuer le *Job*.

Le *Scheduler* permet aux DBA de contrôler différents aspects des planifications tels qu'attribuer des priorités aux jobs.

Ils sont utiles pour assurer la limitation des ressources pendant que des *jobs* s'exécutent.

28.1 Concepts du scheduler

Les concepts avancés du scheduler permettent des contrôles plus poussés des éléments de planification comme par exemple la priorité des jobs.

Les composants sont :

- ♦ Une **JOB CLASS** définit une catégorie de **JOB** qui partage des ressources communes de la même façon. Une **JOB CLASS** regroupe des **JOB** dans des entités plus globales.
- ♦ Un **RESSOURCE CONSUMER GROUP** associé aux **JOB CLASS** définit les ressources qui sont allouées pour les **JOBS** dans ces **JOB CLASS**.
- ♦ Un **RESSOURCE PLAN** permet aux utilisateurs de prioriser les ressources (notamment la CPU) parmi les **RESSOURCES CONSUMER GROUP**.
- ♦ Une **WINDOW** est représentée par un intervalle de temps avec un début et une fin bien déterminée, et est utilisé pour activer différents **RESSOURCE PLAN** à des moments différents. Ceci permet de changer l'allocation de la ressource pendant une période de temps comme l'heure dans la journée ou la période dans l'année.
- ♦ Un **WINDOW GROUP** représente une liste de **WINDOW** et permet la gestion plus facile des **WINDOW**. Une **WINDOW** ou un **WINDOW GROUP** peut être utilisé comme un **SCHEDULE** pour un **JOB** afin d'assurer que le job tourne seulement quand une **WINDOW** et son **RESSOURCE PLAN** associé sont actifs.

Ainsi un groupe de ressources offre un moyen de rassembler les utilisateurs qui partagent les mêmes besoins en terme de ressources machine.

Le package **DATABASE_RESSOURCE_MANAGER** peut être utilisé pour allouer une quantité maximale de CPU utilisable par session, définir le degré de parallélisme maximal, et aussi pour spécifier le nombre de sessions qui peuvent être actives pour un groupe.

Un plan de ressources est élaboré pour des groupes de destinataires et fournit un moyen de définir la façon dont les ressources seront allouées.

Un utilisateur peut être déplacé vers un groupe de priorité moindre pour mettre les ressources à la disposition d'autres sessions. Vous pouvez aussi planifier l'activation ou la désactivation des plans de ressources.



28.2 Privilèges associés au Scheduler

Afin de pouvoir utiliser le *Scheduler*, il est indispensable d'avoir un ensemble de privilèges.

Les privilèges system et objet associés à l'utilisation du *Scheduler* sont :

- ⇒ CREATE [ANY] JOB
- ⇒ EXECUTE ANY PROGRAM
- ⇒ EXECUTE ANY CLASS (Java)
- ⇒ MANAGE SCHEDULER
- ⇒ EXECUTE ON <program or class>
- ⇒ ALTER ON <Job, program or schedule>
- ⇒ ALL ON >Job, program, class or schedule>

Un ensemble de privilèges permettant d'utiliser le *Scheduler* sont disponibles dans le rôle :

⑩ SCHEDULER_ADMIN

- ◆ CREATE JOB
- ◆ CREATE ANY JOB
- ◆ EXECUTE ANY PROGRAM
- ◆ EXECUTE ANY CLASS
- ◆ MANAGE SCHEDULER



Le rôle SCHEDULER_ADMIN est attribué par défaut au rôle DBA, disponible à l'installation d'une base Oracle.
Il est préférable d'utiliser ce rôle pour l'administration.

Le rôle CREATE JOB permet de créer un *job*, un *Scheduler* ou un *program* dans son propre schéma.

Le rôle MANAGE SCHEDULER permet de créer des fenêtres de temps (*Windows*), des classes ou des groupes de fenêtres.

Le privilège EXECUTE permet à des utilisateurs d'avoir le droit d'utiliser un composant du *Scheduler*. Si l'option WITH GRANT OPTION est spécifiée, cet utilisateur pourra à son tour attribuer le privilège. Il s'agit d'un privilège objet.

Pour permettre à un utilisateur d'utiliser tous les *programs* d'un *job*, il doit avoir le privilège GRANT ANY PROGRAM.

```
| Grant execute on calc_stats to charly ;
```



28.2.1 Privilèges utilisateurs

Pour permettre à un utilisateur de modifier un composant d'un *Scheduler* d'un autre schéma, il doit posséder le privilège objet `ALTER` nécessaire pour cet objet.

Il pourra alors **modifier tous les attributs du *job* sauf les attributs suivants** :

- ⇒ `Program_name, program_type, program_action, number_of_arguments`
- ⇒ Modifier le PL/SQL du job



Jobs, programs et schedules sont créés dans le schéma de l'utilisateur.
Jobs, Class, Windows et Window Groups sont créés dans le schéma `SYS`.

Ainsi pour créer un *job* il n'est pas nécessaire d'avoir tous les privilèges concernant le *schedule*, la fenêtre de temps ou le groupe de fenêtres, le privilège `CREATE JOB` suffit. Par contre, il faut préfixer le job par le schéma `SYS`.

Par exemple, si vous créer un job qui utilise la fenêtre `APPL_USER_WINDOWS` dans votre *schedule* et le programme `UPDATE_REPORT_TABS`.

Vous devez avoir le privilège `CREATE JOB`.

Si le programme ne réside pas dans votre schéma, vous devez avoir le privilège objet `EXECUTE` pour le programme `UPDATE_REPORT_TABS` ou le privilège système `EXECUTE ANY PROGRAM`.

Vous devez qualifier la fenêtre par : `schedule => 'SYS. APPL_USER_WINDOWS'`

Si vous créez un job et que vous désirez l'affecter à une classe spécifique, vous devez avoir le privilège objet `EXECUTE` pour la classe de jobs en question ou le privilège système `EXECUTE ANY CLASS`.



Attention le privilège système `EXECUTE ANY CLASS` est à manipuler avec précaution.

28.2.2 Privilèges administrateurs

Pour administrer un *Scheduler* il faut avoir les privilèges suivants :

- ⇒ `CREATE, DROP, ALTER JOB CLASS, WINDOWS` et `WINDOW GROUPS`
- ⇒ `STOP ANY JOB`, avoir la possibilité d'utiliser l'option `FORCE`
- ⇒ `START` ou `STOP WINDOWS` prématurément

Pour attribuer tous les privilèges nécessaires à un utilisateur, il faut avoir le privilège `MANAGE SCHEDULER`. Il est affecté au rôle `SCHEDULER_ADMIN`. Le rôle `SCHEDULER_ADMIN` est attribué par défaut au rôle `DBA` et permet d'effectuer toutes les tâches d'administration.



28.3 Créer et gérer un programme dans un schedule

Si vous avez une procédure appelée MAJ_SCHEMA_STATS qui collecte les statistiques pour un schéma, vous pouvez créer un programme pour appeler cette procédure comme illustré ci-dessous.

```
BEGIN
DBMS_SCHEDULER.CREATE_PROGRAM
(
  PROGRAM_NAME      => 'prog_stats2' ,
  PROGRAM_ACTION    => 'clo.maj_schema_stats' ,
  PROGRAM_TYPE      => 'stored procedure' ,
  ENABLED           => TRUE
) ;
DBMS_SCHEDULER.CREATE_JOB
(
  JOB_NAME          => 'clo.job_stats2' ,
  PROGRAM_NAME      => 'clo.prog_stats2' ,
  START_DATE        => '20-DEC-05' 07.00.00 AM Greenwich' ,
  REPEAT_INTERVAL   => 'FREQ=HOURLY ; INTERVAL=2' ,
  END_DATE          => '20-DEC-06' 07.00.00 AM Greenwich' ,
  COMMENTS          => 'Toutes les 2 heures pendant 1 an'
) ;
END ;
/
```

Ce script crée un job qui fait tourner le programme PROG_STAT2 toutes les 2 heures pendant une année, la date de début étant le 20 décembre 2005 à 7H00.



Pour créer un job qui utilise un programme dans un autre schéma, l'utilisateur qui crée ce job doit avoir le privilège d'accéder au programme.

28.4 Les JOBS

28.4.1 Créer un JOB

Plusieurs étapes sont nécessaires pour créer un job :

Spécifier les composants en utilisant la procédure CREATE_JOB.

Spécifier la tâche à sauvegarder.

Le programme et le schedule à appeler.

Le programme et les spécificités du schedule à appeler.



Exemple

Créer un job qui exécute des scripts de sauvegardes toutes les nuits à 23H00, et qui commence cette nuit.

```
set echo on

BEGIN
DBMS_SCHEDULER.CREATE_JOB
(
  job_name          => 'clo.sauv',
  job_type          => 'executable',
  job_action        => 'd:\admin10\schedul\sauv_23.bat',
  start_date        => 'trunc(sysdate)+23/24',          /*cette nuit à 23H00 */
  repeat_interval    => 'trunc(sysdate+1)+23/24',        /* prochaine nuit à 23H00 */
  comments          => 'sauvegardes toutes les nuits à 23H00'
);
END;
/
```



Le job est créé avec le statut **DISABLED** par défaut.
Il devient actif et utilisable par un *Scheduler* dès qu'il est explicitement **ENABLED**.
Son nom est de la forme [SCHEMA].JOB.

Par défaut, un job est créé dans le schéma courant.

Il est possible de créer un job dans un autre schéma à condition de spécifier le schéma dans lequel il doit être créé.

Le schéma propriétaire est l'utilisateur du job mais le job est exécutable avec les privilèges du propriétaire du job.

L'environnement « NLS » du job (date et heure) est l'environnement de référence pour l'exécution du job.

Le paramètre **JOB_TYPE** indique le type de programme appelé, il peut prendre les valeurs :

- ◆ **PLSQL_BLOCK** : bloc PL/SQL anonym.
- ◆ **STORED_PROCEDURE** : procédure cataloguée PL/SQL, Java ou externe.
- ◆ **EXECUTABLE** : programme exécutable en ligne de commande (dos, unix).

Le paramètre **JOB_ACTION** indique le nom du programme à exécuter. Il dépend du paramètre **JOB_TYPE**.

Le paramètre **REPEAT_INTERVAL** permet de spécifier l'intervalle de réveil du programme appelé en utilisant une expression calendaire ou date, du type **DATE** ou **TIMESTAMP WITH TIME ZONE**.

La méthode de base pour établir la périodicité d'un job se fait en mettant l'attribut **REPEAT_INTERVAL** à la valeur de l'expression calendaire qui possède 3 parties principales :

- ⇒ fréquence (la spécification d'une fréquence est obligatoire)
- ⇒ intervalle de répétition
- ⇒ éléments spécifiques (qui fournissent de l'information détaillée sur la période pendant laquelle le job doit tourner)



Expression datetime :

```
REPEAT_INTERVAL => 'SYSDATE + 36/24'  
REPEAT_INTERVAL => 'SYSDATE + 1'  
REPEAT_INTERVAL => 'SYSDATE + 15/(24*60)' /* 15 minutes */
```

Expression calendaire :

```
REPEAT_INTERVAL => 'FREQ=HOURLY ; INTERVAL=4'  
REPEAT_INTERVAL => 'FREQ=DAILY ;'  
REPEAT_INTERVAL => 'FREQ=MINUTELY ; INTERVAL=5'  
REPEAT_INTERVAL => 'FREQ=YEARLY ;  
BYMONTH=MAR,JUN,SEP,DEC ; /* mois */  
BYMONTHDAY=15' /* N° jour */
```

Si vous voulez créer un *job* qui tourne toutes les 2 semaines ou toutes les 5 minutes ou toutes les secondes, vous utiliserez une combinaison entre la fréquence et l'intervalle.

```
toutes les 2 semaines :      freq=weekly ; interval=2  
toutes les 5 minutes :      freq=minutely ; interval=5  
toutes les secondes :       freq=secondly ;
```

Si vous avez besoin de spécifier des intervalles de définition plus complexes comme le 15^{ème} jour du mois, toutes les 4 semaines le lundi, ou, à 6H23 tous les mardis, il faut utiliser les clauses BY* pour fournir cette information complémentaire.

```
15ème jour du mois :          freq=monthly ; bymonthday=15  
toutes les 4 semaines le lundi : freq=yearly ;  
                                byweekno=4,8,12,16,20,24,28,32,36,40,44,48,52 ;  
                                byday=mon  
à 6H23 tous les mardis :      freq=weekly ; byday=tue ; byhour=6 ; byminute=23
```



Oracle ne garantit pas que le job se déroule à l'heure exacte !
L'exécution se déclenche en fonction de la disponibilité des ressources machine.

28.4.2 Spécifier des schedules pour un job

Le temps pendant lequel un job démarre et se termine est spécifié en utilisant le type de données `TIMESTAMP WITH TIME ZONE`.

La précision d'un *schedule* est la seconde (pas moins). Bien que le type de données `TIMESTAMP WITH TIME ZONE` soit plus précis. Le *Scheduler* arrondit à la seconde supérieure.



La procédure `CREATE_SCHEDULE` du package `DBMS_SCHEDULER` permet de sauvegarder le *schedule* comme un objet de schéma. Ceci permet d'utiliser le même *schedule* par plusieurs `JOB`s ou `WINDOWS`.

Le paramètre `REPEAT_INTERVAL` pour un *schedule* doit être créé en utilisant l'expression calendaire. Vous ne pouvez pas utiliser des expressions `DATETIME` pour spécifier l'intervalle de répétition pour un *schedule* sauvegardé.

Le paramètre `END_DATE` pour un *schedule* sauvegardé correspond à la date après laquelle le *schedule* n'est plus valable.

Le *Scheduler* supporte intégralement toutes les fonctionnalités `NLS` fournies par la base de données.

Par exemple, vous pouvez utiliser tous les paramètres de type `NLS_TIMESTAMP_TZ_FORMAT` ainsi que les types de données `TIMESTAMP WITH TIME ZONE`, par exemple :

```
| 1 :00 :00p.m.  
| 13 :00 :00 hrs  
| 2003-04-15 8 :00 :00 US/Pacific  
| 8 :00 :00 -8 :00  
| 2003-01-31 09 :26 :50.124
```

28.4.3 Créer et utiliser des *schedules*

L'utilisation d'un *schedule* permet de gérer l'exécution planifiée d'une multitude de `JOB`s sans avoir à mettre à jour les définitions de cet ensemble de jobs. En effet, on utilise un *schedule* pour spécifier le temps d'exécution d'un `JOB` au lieu de spécifier le temps d'exécution du job dans la définition de celui-ci.

Si un *schedule* est modifié alors chaque job qui utilise ce *schedule* utilisera automatiquement le nouveau *schedule*.

La procédure `CREATE_SCHEDULE` du package `DBMS_SCHEDULER` permet de créer un *schedule*.

Le paramètre `START_DATE` représente la date à laquelle le *schedule* devient actif. Le *schedule* ne peut pas faire référence à toute autre date avant cette date.

Le *schedule* n'est plus actif après la valeur du paramètre `END_DATE`.

Il est possible de planifier des exécutions répétées en fournissant une expression calendaire pour le paramètre `REPEAT_INTERVAL`. Cette expression calendaire est utilisée pour générer la prochaine date d'exécution.

Les dates après le paramètre `END_DATE` ne seront pas incluses dans le *schedule*.

```
| BEGIN  
| DBMS_SCHEDULER.CREATE_SCHEDULE  
| (  
|   SCHEDULE_NAME => 'schedule_stats' ,  
|   START_DATE   => SYSTIMESTAMP ,  
|   END_DATE     => SYSTIMESTAMP + 30  
| REPEAT_INTERVAL => 'FREQ=HOURLY ; INTERVAL=4' ,  
| COMMENTS      => 'Toutes les 4 heures'  
| ) ;
```



```
DBMS_SCHEDULER.CREATE_JOB  
(  
  JOB_NAME      => 'clo.job_stats' ,  
  PROGRAM_NAME  => 'clo.calc_stats2' ,  
  SCHEDULE_NAME => 'schedule_stats'  
);  
END ;  
/
```

Dans cet exemple, un *schedule* appelé SCHEDULE_STATS est créé avec :

- un intervalle de répétition de 4 heures
- il commence immédiatement
- il dure pendant 30 jours

Le *schedule* sera utilisé par la suite quand le job JOB_STAT sera créé pour déterminer quand le job sera exécuté.

28.5 Les JOB CLASS

28.5.1 Créer une JOB CLASS

La procédure CREATE_JOB_CLASS du package DBMS_SCHEDULER permet la création d'une JOB CLASS.

```
Exemple  
EXECUTE DBMS_SCHEDULER.CREATE_JOB_CLASS  
(  
  JOB_CLASS_NAME      => 'class_jobs' ,  
  RESOURCE_CONSUMER_GROUP => 'group_jobs' ,  
  LOGGING_LEVEL        => DBMS_SCHEDULER.LOGGING_OFF  
);
```

Une fois la JOB CLASS créée, les JOBS qui sont membres de cette JOB CLASS peuvent être spécifiés à la création des JOBS ou après leur création en utilisant la procédure SET_ATTRIBUTE du package DBMS_SCHEDULER.

Le *Scheduler* utilise le concept de JOB CLASS pour gérer l'allocation des ressources pour les différents JOBS. La configuration de l'allocation de ressources est faite par l'attribution d'une JOB CLASS à un CONSUMER GROUP.

Le CONSUMER GROUP auquel une JOB CLASS a été attribuée peut être spécifié au moment de la création d'une JOB CLASS ou bien ultérieurement en utilisant la procédure SET_ATTRIBUTE.



Il existe une JOB CLASS par défaut appelée DEFAULT_JOB_CLASS.



Si un job n'est pas associé à une `JOB CLASS` alors le job appartient à la `JOB CLASSE` par défaut.

Le paramètre `DEFAULT_JOB_CLASS` est associé avec le `RESSOURCE CONSUMER GROUP` par défaut, appelé `DEFAULT_CONSUMER_GROUP`. Cela arrive quand un `RESSOURCE CONSUMER GROUP` n'est pas spécifié à la création d'une `JOB CLASS`.

Les `JOBS` dans la `JOB CLASS` par défaut ou dans une `JOB CLASS` associée au `DEFAULT_CONSUMER_GROUP` peuvent ne pas avoir alloué assez de ressources pour accomplir leur tâche quand le `Ressource Manager` est activé.



Une `JOB CLASS` appartient toujours au schéma `SYS`. La création d'une `JOB CLASS` nécessite le privilège `MANAGE_SCHEDULER`.

28.6 Gestion des Logues de **JOB**

Par défaut tous les `JOBS` sont logués.

- ⇒ A la création d'une nouvelle `JOB CLASS`, il existe des paramètres qui vont contrôler que l'information sera loguée, et la persistance de cette information pourra être spécifiée.

28.6.1 Le package `DBMS_SCHEDULER`

Dans le package `DBMS_SCHEDULER`, le paramètre `LOGGING_LEVEL` pour une `JOB CLASS` peut avoir une des valeurs constantes suivantes :

- ♦ `LOGGING_OFF` : aucun log n'est créé pour tous les jobs de cette class.
- ♦ `LOGGING_RUNS` : le *scheduler* écrit des infos détaillées dans la logue du job pour chaque exécution de chaque job dans cette class.
- ♦ `LOGGING_FULL` : en plus de tracer chaque exécution dans la logue du job, le *scheduler* logue aussi toute autre opération exécutée pour tous les `JOBS` de cette `JOB CLASS`. Par exemple, la création de nouveaux `JOBS`, l'activation ou la désactivation de nouveau `JOBS` etc...



La vue `DBA_SCHEDULER_JOB_LOG` stocke une ligne pour chaque opération « loguée » du `JOB`.

Le paramètre `LOG_HISTORY` spécifie combien de jours une entrée de la logue reste dans le fichier de log avant d'être supprimée.



Le job `PURGE_LOG` est créé automatiquement. Il supprime les anciennes entrées de logues une fois par jour.

Ces entrées peuvent aussi être supprimées manuellement en utilisant la procédure :

`DBMS_SCHEDULER.PURGE_LOG.`

```
EXECUTE DBMS_SCHEDULER.CREATE_JOB_CLASS
(
  JOB_CLASS_NAME => 'class_jobs' ,
  RESOURCE_CONSUMER_GROUP => 'Group_jobs' ,
  LOGGING_LEVEL => DBMS_SCHEDULER.LOGGING_RUNS ,
  LOG_HISTORY => 30
);
```

28.6.2 Les logs des JOBS

La vue `DBA_SCHEDULER_JOB_LOG` affiche une ligne pour chaque opération effectuée par le `JOB` pendant son exécution.

```
SELECT job_name, operation, owner
FROM DBA_SCHEDULER_JOB_LOG
/
```

Vous pouvez purger les logs de `JOBS` :

- ⇒ Automatiquement à travers la valeur du paramètre `PURGE_LOG`.
Le paramètre `PURGE_LOG` définit les conditions de purge des logs.
Identique à l'utilisation de la procédure `DBMS_SCHEDULER.PURGE_LOG`.

```
EXEC DBMS_SCHEDULER.PURGE_LOG(
  Log_history => 1,
  Job_name => 'DEV_TEST_JOB1' ) ;
```

La durée de purge par défaut est 30 jours, mais elle peut être modifiée en utilisant le paramètre `LOG_HISTORY`, les valeurs qui peuvent être utilisées vont de 1 à 999.

Par exemple, pour spécifier la durée de purge de 60 jours pour le job `APPL_JOB_CLASS`, il suffit d'utiliser la procédure :

```
EXEC DBMS_SCHEDULER.SET_ATTRIBUTE(
  'APPL_JOB_CLASS', 'log_history', '60' ) ;
```

La procédure `DBMS_SCHEDULER.PURGE_LOG` permet d'effectuer des purges manuellement.

Elle contient les paramètres suivants :

- ♦ `LOG_HISTORY` : spécifie la durée de conservation, cette valeur va de 0 à 999. Si la valeur « 0 » est positionnée, alors tous les `LOGs` sont purgés.
- ♦ `WHICH_LOG` : définit le `JOB` ou la fenêtre à supprimer. Ces valeurs sont `JOB_LOB`, `WINDOW_LOG` et `JOB_AND_WINDOWS_LOG`.
- ♦ `JOB_NAME` : précise le nom du `JOB` des `LOGs` à purger. Vous pouvez spécifier une liste de noms de *Jobs* ou de *Job Classes*, séparés par des virgules.



28.7 Les Windows

28.7.1 Créer une WINDOW

Le but d'une WINDOW est de changer le RESSOURCE PLAN qui est actif pour une période spécifique.

La WINDOW est représentée par un intervalle de temps comme par exemple « chaque jour de 8H00 à 18H00 ». Ce type de WINDOW est paramétré comme un *schedule* qui spécifie un modèle de date de démarrage et une durée (en minutes).

Un *ressource plan* valable pour tout le *system* peut être associé avec une *Windows* pour gérer l'utilisation globale des ressources pour les jobs et les sessions qui tournent dans cette *Window*.

Par exemple, la priorité des JOBS change après une certaine période de temps, ainsi il peut être important dans certaines situations de charger des jobs qui tournent la nuit et de leur allouer un pourcentage important des ressources de la base de données.

Pendant la journée, les jobs applicatifs sont plus importants et doivent se voir allouer un pourcentage plus important de ressources.

Pour réaliser ceci, vous devez changer le *ressource plan* en utilisant un *schedule*.

⇒ le concept de WINDOW vous permet de le faire.



Pour créer une *Window*, il faut avoir le privilège system `MANAGE SCHEDULER`.
Les *Windows* sont créées dans le schéma `SYS`.

Une *Window* est « ouverte » pendant une période de temps définie.

Le paramètre `DURATION` spécifie combien de temps cette *Window* restera ouverte.

La durée est spécifiée comme une donnée de type `INTERVAL DAY TO SECOND`.

Pour savoir quand rouvrir la *Window*, Oracle utilise la valeur du paramètre `REPEAT_INTERVAL` qui sera évalué par rapport au paramètre `START_DATE`.

La priorité est spécifiée en utilisant le paramètre `WINDOW_PRIORITY`, la valeur par défaut est `LOW_PRIORITY`.

```
BEGIN
DBMS_SCHEDULER.CREATE_WINDOW
(
WINDOW_NAME => 'dec_nuit' ,
RESOURCE_PLAN => 'fin_annee' ,
START_DATE  => '01-DEC-05 06.00.00 PM EST' ,
REPEAT_INTERVAL => 'FREQ=DAILY; BYHOUR=18' ,
DURATION      => '0 12/000/00' ,
END_DATE      => '31-DEC-05 06.00.00 AM EST' ,
COMMENTS      => 'Every day at 6:00 PM'
) ;
END;
/
```

La *Window* devient active à 6H00 le 1^{er} décembre 2005.



Le paramètre `DURATION` spécifie combien de temps cette *Window* restera ouverte. La durée est spécifiée comme une donnée de type `INTERVAL DAY TO SECOND`. La valeur « 0 12 :00 :00 » représente 0 jours 12 heures 0 minutes et 0 secondes. Ceci signifie que la *WINDOW* ferme à 6 heures le 2 décembre 2003.

La prochaine fois que la *WINDOW* sera ouverte est calculée en utilisant la valeur du `REPEAT_INTERVAL` qui sera évalué par rapport à 6 heures le 2 décembre 2005. A 6h00 le 31 décembre 2005 la *WINDOW* sera fermée et désactivée. Elle ne sera plus ouverte par la suite.

Pendant que la *Window* appelée `DEC_NUIT` sera ouverte les ressources allouées pour les jobs seront déterminées par les éléments spécifiés dans le ressource plan appelé `FIN_ANNEE`. Il n'y a aucune priorité spécifiée pour cette *Window*.



La priorité des *Windows* est significative quand plusieurs *Windows* définissent la même période.

28.7.2 Attribuer des priorités aux JOBS dans les WINDOWS

Les `JOBS CLASS` sont utilisées pour catégoriser les jobs.

Une `JOB CLASS` est associée à un *Ressource Consumer group*.

Les *ressources plan* actifs déterminent les ressources allouées à chaque *Resource Consumer Group* et par là même à chaque *job class*.

Lorsque vous créez plusieurs *jobs* dans la base de données, vous devez spécifier quels *jobs* ont la priorité la plus haute.

Pour une *Window* en particulier, vous pouvez faire tourner plusieurs *jobs*, chacun ayant sa propre priorité.

Ainsi, il est possible d'avoir différents niveaux de classe ou de job.

- ⇒ La première priorité est le niveau de la classe dans le *Ressource Plan*.
- ⇒ La seconde priorité est la priorité du job dans la `JOB CLASS`.



Le niveau de priorité est approprié quand deux jobs de la même classe sont supposés démarrer au même moment. La priorité n'est pas gérée entre des jobs de différentes `JOB CLASS`.

Pour définir la priorité, il faut utiliser la procédure `SET_ATTRIBUTE` du package `DBMS_SCHEDULER`.

Les priorités sont définies en utilisant des rangs allant de 1 à 5. Le rang 1 définit la priorité la plus haute.



```
BEGIN
DBMS_SCHEDULER.SET_ATTRIBUTE
(
NAME    => 'job3',
ATTRIBUTE => 'job_priority' ,
VALUE    => 2
);
END;
/
```



Si aucune priorité n'est affectée au job, le rang 3 est affecté par défaut.
Pour afficher la priorité en cours des différents jobs, utiliser la requête :

```
SELECT JOB_NAME, JOB_PRIORITY
FROM    DBA_SCHEDULER_JOBS ;
```

28.8 Activer un composant du scheduler

Pour activer un composant du *Scheduler* utiliser la procédure `ENABLE` du package `DBMS_SCHEDULER`, que ce soit un *Job*, un *Program* ou une *Window*.

```
Exemple
• activer le programme stats3
EXEC DBMS_SCHEDULER.ENABLE( 'clo.stats3' ) ;
• désactiver le job job_stat du schema clo
EXEC DBMS_SCHEDULER.DISABLE( 'clo.job_stat' ) ;
```

De la même façon, pour désactiver un composant du Scheduler, utilisez la procédure `DISABLE` du package `DBMS_SCHEDULER`.

28.9 Gérer des composants du Scheduler

28.9.1 Gérer un JOB

Pour exécuter un Job, utilisez la procédure `RUN_JOB` du package `DBMS_SCHEDULER`. Le Job s'exécute immédiatement dans votre session.

Le Job s'exécute dans votre session comme un Job esclave au lieu d'être exécuté par le Job Coordinateur.

De la même façon, pour arrêter l'exécution d'un Job, utilisez la procédure `STOP_JOB`. L'arrêt du job détruit sa définition.

Cette procédure possède deux arguments :

- ⇒ `JOB_NAME` : le nom du job.
- ⇒ `FORCE` : définit la méthode avec laquelle le Job est arrêté.
- ⇒ Si elle est à `FALSE` (par défaut), le *Scheduler* essaie de terminer le Job proprement. S'il échoue, une erreur est retournée.
- ⇒ Si elle est à `TRUE`, le Job s'arrête brutalement.





Pour utiliser le paramètre `FORCE`, il faut avoir le privilège système `MANAGE SCHEDULER`.

Pour supprimer un Job, utilisez la procédure `DROP_JOB` du package `DBMS_SCHEDULER`. Cette procédure possède deux arguments :

- ⇒ `JOB_NAME` : le nom du job.
- ⇒ `FORCE` : supprime le Job même s'il est en train de s'exécuter. Positionné à `FALSE` par défaut.

Si l'argument `FORCE` est positionné à `TRUE`, tous les *Jobs* en cours d'exécution sont supprimés après avoir été arrêtés.

Si l'on spécifie le nom d'une `JOB CLASS` à la place du nom d'un Job, alors la totalité des Jobs contenus dans la `JOB CLASS` sont supprimés, mais pas la `JOB CLASS` elle-même.

```
•      exécuter un job
DBMS_SCHEDULER.RUN_JOB('clo.job3') ;
•      arrêter un job
DBMS_SCHEDULER.STOP_JOB('clo.job3') ;
•      supprimer des jobs en cours d'exécution
DBMS_SCHEDULER.DROP_JOB('clo.job3, clo.job5, sys.jobclass2') ;
```

28.9.2 Gérer un PROGRAM

Vous pouvez utiliser les procédures `ENABLE` et `DISABLE` du package `DBMS_SCHEDULER` pour activer ou désactiver un composant du scheduler.

Pour supprimer un programme, utilisez la procédure `DROP_PROGRAM` du package `DBMS_SCHEDULER`.

Cette procédure contient deux arguments :

- ⇒ `PROGRAM_NAME` : nom du programme.
- ⇒ `FORCE` : argument permettant de forcer la suppression des programmes.
- ⇒ positionné à `TRUE`, tous les jobs qui référencent ce programme sont désactivés avant la suppression du programme.
- ⇒ positionné à `FALSE` (par défaut), les jobs référençant le programme à supprimer doivent être désactivés un par un avant de pouvoir supprimer le programme.

```
•      activer un programme
EXECUTE DBMS_SCHEDULER.ENABLE('clo.prog1', 'clo.prog2');
•      désactiver un programme
EXECUTE DBMS_SCHEDULER.DISABLE('clo.prog1', 'clo.prog2');
•      supprimer un programme
EXECUTE DBMS_SCHEDULER.DROP_PROGRAM(-
PROGRAM_NAME => 'clo.prog1', -
FORCE       => TRUE);
```



28.9.3 Gérer un schedule

Un *schedule* peut être créé, modifié ou supprimé en utilisant les procédures adéquates :

- | | | |
|------------------|---|-----------------|
| ♦ Pour créer | ⇒ | CREATE_SCHEDULE |
| ♦ Pour modifier | ⇒ | SET_ATTRIBUTE |
| ♦ Pour supprimer | ⇒ | DROP_SCHEDULE |

La procédure SET_ATTRIBUTE permet de modifier les attributs du *schedule*.

La suppression de plusieurs *schedules* est possible en listant leur nom dans les arguments de la procédure DROP_SCHEDULE.

Si des jobs ou des Windows utilisent le *schedule* qui doit être supprimé, il faut alors forcer la suppression en utilisant le paramètre FORCE. Dans ce cas, les *jobs* ou les *Windows* sont désactivés (DISABLE) avant la suppression du *schedule*.



Si vous n'êtes pas propriétaire du *schedule*, vous devez posséder le privilège ALTER ou CREATE ANY JOB.

- modifier l'heure de démarrage du schedule

```
EXECUTE DBMS_SCHEDULER.SET_ATTRIBUTE(-
NAME => 'clô.schedul3', -
ATTRIBUTE => 'start_date',
VALUE      => '01-JAN-2005 9:00:00
```

28.9.4 Gérer une fenêtre de temps : Window

Seule une Window peut être active à un moment donné. Lorsque le temps de démarrage d'une fenêtre est atteint, la fenêtre de temps Window s'ouvre automatiquement.

La procédure OPEN_WINDOW du package DBMS_SCHEDULER permet d'ouvrir une fenêtre prématurément. L'ouverture d'une *Window* ferme n'importe quelle autre Window ouverte, même si la *Windows* a une priorité plus importante car l'attribut FORCE est positionné à TRUE.

L'intervalle de temps spécifié pour la *window* n'est pas altéré, elle s'ouvrira donc normalement la fois suivante.

On peut modifier les paramètres de durée d'ouverture d'une fenêtre déjà ouverte à une durée plus importante. Les nouvelles valeurs sont prises en compte immédiatement.



```
-- ouverture de la window Win_nuit
DBMS_SCHEDULER.OPEN_WINDOW
(
    WINDOW_NAME => 'Win_nuit' ,
    DURATION    => '1 0:00:00' ,
) ;

• fermeture de la window win_nuit
DBMS_SCHEDULER.CLOSE_WINDOW ( 'Win_nuit' ) ;
```

Pour fermer une fenêtre prématurément, il faut utiliser la procédure `CLOSE_WINDOW` du package `DBMS_SCHEDULER`. Les jobs sont arrêtés lorsque la *Window* se ferme s'ils contiennent le paramètre `stop_on_window_close` positionné à `TRUE`. Autrement, ils continuent de s'exécuter.

Par contre, comme le plan de ressources change à cause de la fermeture de la *Window*, les ressources affectées aux jobs changent également.

Une *Window* peut être positionnée à `DISABLE` si elle n'est pas référencée par un job ou si elle est fermée.

Pour rendre inactive une *Window*, il faut positionner le paramètre `FORCE` à `TRUE` en utilisant la procédure `DISABLE` du package `DBMS_SCHEDULER`.

La procédure `DROP_WINDOW` du package `DBMS_SCHEDULER` permet de supprimer une *Window*. La suppression d'une *Window* entraîne la désactivation de tous les jobs référencés par cette *Window*.

Pour supprimer une *Window* active, il faut positionner le paramètre `FORCE` à `TRUE`. La *Window* est alors fermée puis supprimée. Cette *Window* est alors supprimée de tous les *Windows Groups*.

```
DBMS_SCHEDULER.DISABLE
(
    WINDOW_NAME => 'sys.Win_nuit' ,
    FORCE       => TRUE
) ;

DBMS_SCHEDULER.DROP_WINDOW
(
    WINDOW_NAME => 'Win_nuit, Win_mois' ,
    FORCE       => TRUE
) ;
```



Pour ouvrir, fermer, désactiver ou supprimer une *Window*, il faut avoir le privilège système `MANAGE_SCHEDULER`. Comme la *Window* est dans le schéma `SYS` il faut la préfixer par `[SYS.]`.

28.9.5 Priorité des Windows

Le *Scheduler* ne valide pas les dates de démarrage et de fin gérées dans les *Windows*.

Il est donc possible de créer des *Windows* avec des programmes de recouvrement.



Les *Schedulers* ne sont pas arrêtés si la *Window* se ferme, sauf si le paramètre `STOP_ON_CLOSE` est positionné à `TRUE`.

Les règles concernant ces recouvrements sont définies comme suit :

- ⇒ Si des fenêtres de même priorité se chevauchent, la fenêtre qui est active reste ouverte. Cependant si le chevauchement se fait avec une fenêtre d'une priorité plus élevée, la fenêtre de plus faible priorité se ferme et la fenêtre de priorité plus élevée s'ouvre.
- ⇒ Si à la fin d'une fenêtre, il y a plusieurs fenêtres définies, la fenêtre qui a le pourcentage de temps le plus élevé s'ouvre.
- ⇒ Une fenêtre ouverte qui est supprimée est automatiquement fermée.

28.9.6 Gérer les attributs des composants du scheduler

La procédure `SET_ATTRIBUTE` modifie les attributs d'un composant du *Scheduler*. Elle peut accepter des valeurs de plusieurs `DATA TYPE`.

```
BEGIN
DBMS_SCHEDULER.SET_ATTRIBUTE (
Name      => 'MAX_FAILURES',
Attribute => 'MAX_FAILURES',
Value     => 3 );
END;
/
```

Chaque composant de *Scheduler* possède différents attributs qui peuvent être modifiés. Par exemple, il est possible de modifier le `PROGRAM_TYPE` d'un programme, par contre cet attribut n'est pas adapté à un `JOB` ou à une *Window*. De la même façon, il est possible de modifier le `JOB_CLASS` d'un `JOB`.

Pour positionner un attribut à la valeur `NULL`, utilisez la procédure `SET_ATTRIBUTE_NULL`. Cette procédure peut être utilisée pour initialiser tous les attributs.

```
EXEC DBMS_SCHEDULER.SET_ATTRIBUTE_NULL (
'CHARLY.CALC_STATS', 'COMMENTS' );
```

Il existe 3 attributs qui peuvent être gérés au niveau global :

`DEFAULT_TIMEZONE` : utilisé par les `JOBS` et les *Windows* qui emploient une syntaxe calendaire pour leurs intervalles de répétition. Elle est utilisée pour `START_DATE` mais si la date n'est pas fournie, le fuseau horaire est recherché à partir de cet attribut.

`MAX_JOB_SLAVE_PROCESSES` : permet de placer un nombre maximum des processus définis pour une configuration et un chargement particulier. Même si le *Scheduler* détermine automatiquement le nombre de processus optimal, vous pouvez l'utiliser pour fixer une limite pour le *Scheduler*.

`LOG_HISTORY` : détermine le nombre de jours maintenus pour un `JOB` ou une *Window* avant une purge automatique. La valeur par défaut est 30.

```
EXEC DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE (
Attribute => 'log_history',
Value     => '14' );
```





La modification de la valeur d'un attribut prend effet immédiatement, même si le résultat ne se voit pas tout de suite.

La procédure `GET_SCHEDULER_ATTRIBUTE` du package `DBMS_SCHEDULER` permet de retrouver les valeurs des différents attributs.

28.10 Vues du dictionnaire de données

Pour retrouver les valeurs des attributs d'un composant de *Scheduler*, interroger les vues :

- `*_SCHEDULER_JOBS`
- `*_SCHEDULER_PROGRAMS`
- `DBA_SCHEDULER_SCHEDULE`
- `DBA_SCHEDULER_WINDOWS`

```
SELECT max_failures, job_priority, schedule_limit, logging_level
FROM DBA_SCHEDULER_JOBS
WHERE job_name = 'GET_STATS'
AND job_creator = 'CHARLY'
/
```

Les vue du dictionnaire de données permettant de visualiser des informations concernant le Scheduler ou ses composants sont :

- `DBA_SCHEDULER_JOBS` : informations sur les jobs
- `DBA_SCHEDULER_JOBS_ARGS` : liste des arguments d'un job
- `DBA_SCHEDULER_RUNNING_JOBS` : Jobs en train de s'exécuter
- `DBA_SCHEDULER_JOBS_LOG` : LOGs des jobs.
- `DBA_SCHEDULER_JOB_RUN_DETAILS` : informations détaillées sur les jobs
- `DBA_SCHEDULER_PROGRAMS` : informations sur les programmes
- `DBA_SCHEDULER_PROGRAMS_ARGS` : informations sur les arguments des programmes
- `DBA_SCHEDULER_JOBS_CLASSES` : informations sur les Job Classes
- `DBA_SCHEDULER_WINDOWS` : informations sur les Windows
- `DBA_SCHEDULER_WINDOW_DETAILS` : informations détaillées sur les Windows
- `DBA_SCHEDULER_WINDOW_LOG` : informations sur les Logs des Windows

La vue `DBA_SCHEDULER_JOB_LOG` stocke une ligne pour chaque opération « *loguée* » du `JOB`.

La vue `DBA_SCHEDULER_JOB_RUN_DETAILS` affiche une ligne par `JOB` en cours d'exécution.

Chacune des lignes contient des informations concernant l'exécution du `JOB` dans l'instance.



DBA_SCHEDULER	
LOG_ID	identifiant de la logue en entrée
LOG_DATE	date de la logue
OWNER	Propriétaire du JOB
JOB_NAME	Nom du Job
STATUS	Numéro de la première erreur rencontrée
ERROR#	Plus grand numéro SCN écrit dans l'archive
REQ_START_DATE	Date à laquelle le JOB a été programmé par le Scheduler
ACTUAL_START_DATE	Date de début d'exécution
RUN_DURATION	Durée de l'exécution du JOB
INSTANCE_ID	Identifiant de l'instance
SESSION_ID	Identifiant de la session
SLAVE_PID	Identifiant du processus qui exécute le JOB
CPU_USED	Quantité de CPU utilisée par l'exécution du JOB
ADDITIONAL_INFO	Informations supplémentaires concernant l'exécution du JOB

```
• Visualiser le statut d'un job
--
SELECT job_name, program_name, job_type, state
FROM DBA_SCHEDULER_JOBS
WHERE owner = 'CHARLY'
/

• Visualiser l'instance dans laquelle s'exécute le job
--
SELECT owner, job_name, running_instance, resource_consumer_group
FROM DBA_SCHEDULER_RUNNING_JOBS
WHERE owner = 'CHARLY'
/

• Visualiser les arguments d'un job
--
SELECT value
FROM DBA_SCHEDULER_JOB_ARGS
WHERE owner = 'CHARLY'
AND job_name = 'MON_JOB'
/

• Auditer l'activité des jobs
--
SELECT owner, job_name, job_class, operation, status
FROM DBA_SCHEDULER_JOB_LOG
/
```



- Visualiser les différents status d'un job pendant son exécution

```
--  
SELECT job_name, status, error#, run_duration, cpu_used  
FROM DBA_SCHEDULER_JOB_RUN_DETAILS  
/  
  
• Visualiser les informations des programmes qui s'exécutent

```
--
SELECT program_name, program_type, program_action
FROM DBA_SCHEDULER_PROGRAMS
/

```


```



29 Jeux de caractères et paramètres NLS

29.1 Introduction

NLS a pour fonction d'adapter automatiquement à la langue locale les utilitaires de base de données et les messages d'erreur, l'ordre de tri, la date, l'heure, les conventions monétaires, numériques et calendaires.

Les opérations liées à la langue sont gérées par un certain nombre de paramètres coté client et coté serveur.

Le serveur et le client peuvent se trouver à des emplacements différents.

Au cas où chacun d'entre eux utilise des caractères différents, ORACLE fait automatiquement la conversion.

Caractéristiques du NLS :

- ⇒ Prise en charge de la langue
- ⇒ Prise en charge du territoire
- ⇒ Prise en charge du jeu de caractères
- ⇒ Tri linguistique
- ⇒ Prise en charge des messages
- ⇒ Formats date et heure
- ⇒ Formats numériques
- ⇒ Formats monétaires

Jeux de caractères de la base et jeu de caractères national

Le jeu de caractères est créé à la création de la base de données par la commande :

```
CREATE DATABASE ...    clause CHARACTER SET  
                      clause NATIONAL CHARACTER SET
```

Les jeux de caractère de la base de données et le jeu de caractères national (client + serveur) doivent être très proches.

Une base oracle possède 2 jeux de caractères :

- ⇒ Jeu de caractères standard : Pour les types SQL : CHAR, VARCHAR et LOB
- ⇒ Jeu de caractères national : Pour les types SQL : NCHAR, NVARCHAR et NLOB

A partir de la version 9i, le jeu de caractères national doit impérativement être un jeu de caractères UNICODE.

- ⇒ 2 valeurs possibles : UTF8 et AL16UTF16



La commande ALTER SESSION permet de modifier le comportement de la session en cours, on peut changer les caractères NLS de la session :

- ♦ ALTER SESSION SET NLS_LANGUAGE=' FRENCH' ;
- ♦ ALTER SESSION SET NLS_TERRITORY=' FRANCE' ;
- ♦ ALTER SESSION SET NLS_DATE_FORMAT= »DD.MM.RRRR « ;
- ♦ ALTER SESSION SET NLS_TIMESTAMP_FORMAT="DD.MM.RRRR HH24:MI:SSXFF" ;
- ♦ ALTER SESSION SET NLS_LANGUAGE=FRENCH_FRANCE.WE8MSWIN1252 ;

La globalisation NLS (National Language Support) permet

- ♦ le support du traitement des données dans les différentes représentations de caractères utilisés par le matériel
- ♦ La transparence de la différence des jeux de caractères entre le serveur et le client
- ♦ Le support d'opérations dépendantes de la langue de l'utilisateur final permettant de les spécifier par session : messages du serveur, format des dates et des nombres, ou encore tris alphabétiques
- ♦ La variable d'environnement NLS_LANG qui définit l'encodage de caractères d'un terminal client :
 - Les données transmises entre le client et le serveur sont automatiquement converties
 - L'encodage de la base de données doit être un ensemble de niveau supérieur ou équivalent pour tous les encodages clients

Si le jeu de caractères du client est différent de celui du serveur alors une conversion est opérée dans les 2 sens, mais il est conseillé d'avoir le même jeu de caractères sur le client et le serveur, car si un caractère utilisé n'a pas de correspondant dans le jeu de caractères en face, alors une perte d'information est inévitable.

La variable d'environnement NLS_LANG sur le client joue un rôle déterminant dans la conversion des caractères.

Le jeu de caractères national a été ajouté à partir de la version 8i.

Une base oracle possède 2 jeux de caractères :

- ⇒ **Jeu de caractère standard**
Pour les types SQL : CHAR, VARCHAR et LOB
- ⇒ **Jeu de caractères national**
Pour les types SQL : NCHAR, NVARCHAR et NLOB

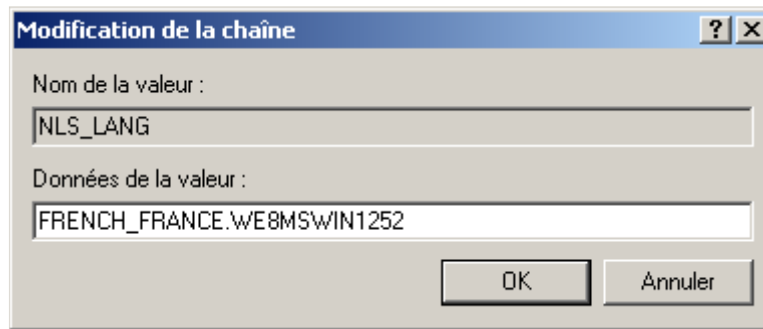
A partir de la version 9i, le jeu de caractères national doit impérativement être un jeu de caractères UNICODE.

- ⇒ 2 valeurs possibles : UTF8 et AL16UTF16

La globalisation NLS (National Language Support) permet :

- ♦ le support du traitement des données dans les différentes représentations de caractères utilisés par le matériel.
- ♦ La transparence de la différence des jeux de caractères entre le serveur et le client
- ♦ Le support d'opérations dépendantes de la langue de l'utilisateur final permettant de les spécifier par session : messages du serveur, format des dates et des nombres, ou encore tris alphabétiques.





Paramètres par défaut :

23/05/08
page: 1

OPTIONS PAR DEFAULT DE LA BASE DE DONNEES

Option	Valeur
-----	-----
DICT.BASE	2
DEFAULT_TEMP_TABLESPACE	TEMP
DEFAULT_PERMANENT_TABLESPACE	USERS
DEFAULT_TBS_TYPE	SMALLFILE
NLS_LANGUAGE	AMERICAN
NLS_TERRITORY	AMERICA
NLS_CURRENCY	\$
NLS_ISO_CURRENCY	AMERICA
NLS_NUMERIC_CHARACTERS	.,
NLS_CHARACTERSET	WE8MSWIN1252
NLS_CALENDAR	GREGORIAN
NLS_DATE_FORMAT	DD-MON-RR
NLS_DATE_LANGUAGE	AMERICAN
NLS_SORT	BINARY
NLS_TIME_FORMAT	HH.MI.SSXF AM
NLS_TIMESTAMP_FORMAT	DD-MON-RR HH.MI.SSXF AM
NLS_TIME_TZ_FORMAT	HH.MI.SSXF AM TZR
NLS_TIMESTAMP_TZ_FORMAT	DD-MON-RR HH.MI.SSXF AM TZR
NLS_DUAL_CURRENCY	\$
NLS_COMP	BINARY
NLS_LENGTH_SEMANTICS	BYTE
NLS_NCHAR_CONV_EXCP	FALSE
NLS_NCHAR_CHARACTERSET	AL16UTF16
NLS_RDBMS_VERSION	10.2.0.1.0
GLOBAL_DB_NAME	NSKEPP.REGRESS.RDBMS.DEV.US.ORACLE.COM
EXPORT_VIEWS_VERSION	8
DBTIMEZONE	00:00

Extrait du fichier de paramètres :

```
nls_calendar
nls_comp
nls_currency
nls_date_format
nls_date_language
nls_dual_currency
nls_iso_currency
nls_language
nls_length_semantics
nls_nchar_conv_excp
nls_numeric_characters
nls_sort
nls_territory
nls_time_format
nls_timestamp_format
nls_timestamp_tz_format
nls_time_tz_format
```



Certains paramètres influencent l'utilisation des index par l'optimiseur :

NLS_SORT : détermine le traitement des chaînes de caractères dans les tris :

- ♦ Pas d'effet sur les tris des index
- ♦ Pas d'effet sur les tris internes d'oracle (suppressions des doublons par exemple)

NLS_COMP : détermine le comportement lors des opérations de comparaison des chaînes de caractères :

- ♦ BINARY
- ♦ ANSI
- ♦ LINGUISTIC

29.2 Migration de jeux de caractères

Faites attention lors de la migration de jeux de caractères car une colonne définie avec une longueur convenable dans un jeu de caractères peut être tronquée dans un autre jeu de caractères.

Il faut vérifier que le paramètre NLS_LANG de la base a le même jeu de caractères du système d'exploitation du client.

⇒ **Faire une sauvegarde de la base avant une migration de jeu de caractères**

L'outil CSSCAN exécuté sous le système d'exploitation permet d'afficher les problèmes possibles de la base de données à convertir. Le résultat des problèmes affichés par l'outil peut nécessiter l'intervention du support Oracle.

L'installation de l'outil *csscan* nécessite l'exécution sous SYS du script :

⇒ `~/rdbms/admin/csminst.sql`

```
C:\oracle>csscan help=y
Character Set Scanner v2.1 : Release 10.2.0.0.0 - Production on Lun. Nov. 10 12:
12:51 2008
Copyright © 1982, 2005, Oracle. All rights reserved.
You can let Scanner prompt you for parameters by entering the CSSCAN
command followed by your username/password:

Example: CSSCAN SYSTEM/MANAGER
Or, you can control how Scanner runs by entering the CSSCAN command
followed by various parameters. To specify parameters, you use keywords:

Example: CSSCAN SYSTEM/MANAGER FULL=y TOCHAR=utf8 ARRAY=1024000 PROCESS=3
Keyword      Default Prompt Description
-----
USERID              yes  username/password
FULL              N    yes  scan entire database
USER              yes  owner of tables to be scanned
TABLE             yes  list of tables to scan
COLUMN            yes  list of columns to scan
EXCLUDE           yes  list of tables to exclude from scan
TOCHAR            yes  new database character set name
FROMCHAR          yes  current database character set name
TONCHAR           yes  new national character set name
FROMNCHAR         yes  current national character set name
ARRAY            1024000 yes  size of array fetch buffer
PROCESS           1    yes  number of concurrent scan process
```




```
MAXBLOCKS          split table if block size exceed MAXBLOCKS
CAPTURE            N          capture convertible data
SUPPRESS           maximum number of exceptions logged for each table
FEEDBACK           report progress every N rows
BOUNDARIES         list of column size boundaries for summary report
LASTRPT           N          generate report of the last database scan
LOG                scan      base file name of report files
PARFILE            parameter file name
PRESERVE           N          preserve existing scan results
LCSD               N          no enable language and character set detection
LCSDDATA           LOSSY     no define the scope of the detection
HELP              N          show help screen (this screen)
QUERY             N          select clause to scan subset of tables or columns
-----
Scanner terminated successfully.
C:\oracle>
```

La migration peut se faire par EXPORT/IMPORT ou en utilisant le script CSALTER, à condition que le nouveau jeu de caractères soit un sur-ensemble de l'ancien.

Le script csalter.plb remplace l'instruction SQL :

⇒ **ALTER DATABASE CHARACTER SET nouveau_jeu_de_caractères ;**

29.2.1 Migration du jeu de caractères par EXPORT/IMPORT

Vérifiez la convertibilité du jeu de caractères avec CSSCAN. En effet, celui-ci peut rapporter un problème de troncature de certaines colonnes de la base.

Mode opératoire :

- ⇒ Exportation de la base de données.
- ⇒ Création d'une nouvelle base de données dans le jeu de caractères désiré.
- ⇒ Recréation de tables si nécessaire avec des colonnes plus grandes pour les données tronquées
- ⇒ IMPORT des données dans la nouvelle base de données.



la variable NLS_LANGS doit avoir une valeur qui correspond au jeu de caractères de la base source dans les 2 phases.



Vues du dictionnaire de données

Les vues du dictionnaire de données intéressantes sont :

- DATABASE_PROPERTIES : informations sur les propriétés par défaut de la base de données
- NLS_DATABASE_PARAMETERS : valeurs par défaut des paramètres NLS utilisés par la base de données (inclus les 2 jeux de caractères et la version de la base).
- NLS_INSTANCE_PARAMETERS : valeurs des paramètres utilisés par l'instance.
- NLS_SESSION_PARAMETERS : valeurs des paramètres utilisés par la session.
- V\$NLS_PARAMETERS : valeurs des paramètres utilisés par la session (incluent les 2 jeux de caractères de la base)
- • V\$NLS_VALID_VALUE : liste des valeurs valident pour certains paramètres.



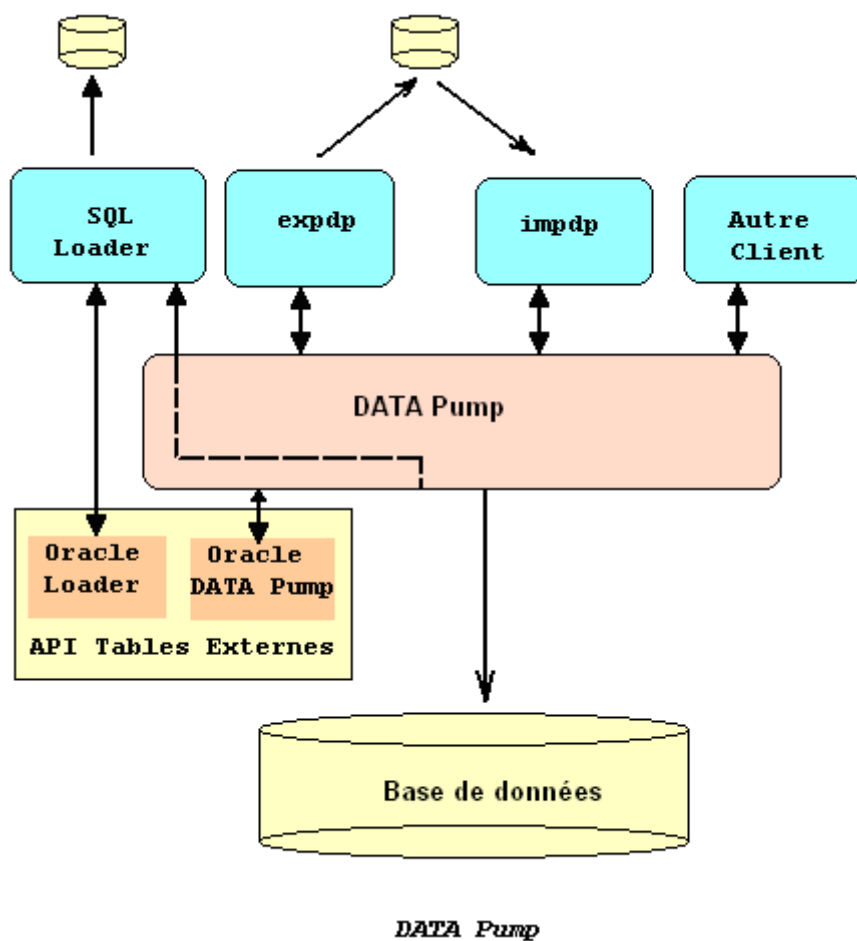
30 Présentation de l'utilitaire DATA Pump

Data pump est un nouvel outil qui permet de charger ou décharger des données à grande vitesse.

Il peut être appelé *via* le package PL/SQL, `DBMS_DATAPUMP`.

Oracle 10g introduit des nouveaux outils :

- ⇒ De nouvelles commandes pour Export et Import appelées respectivement `EXPDP` et `IMPDP`.
- ⇒ Une interface Web d'Import et d'Export accessible à partir du Database Control.



Les composants du DATA Pump sont :

- ◆ Direct Path API (DPAPI) : Oracle 10g supporte une interface de chemin directe qui minimise la conversion et le processus des données au moment du chargement et du déchargement des données.
- ◆ Services des Tables Externes : DATA Pump utilise le nouveau Driver d'accès ORACLE_DATAPUMP qui fournit des accès en lecture et écriture aux tables externes à des fichiers qui contiennent des chaînes de caractères binaires.
- ◆ Le package DBMS_METADATA est utilisé par les processus d'exécution pour tout chargement ou déchargement des métas-données (CLOB, BLOB). Les définitions des objets de la base sont stockées en utilisant XML plutôt que SQL.
- ◆ Le package DBMS_DATAPUMP inclut l'API pour les fonctions d'Import et Export rapides, ainsi que le déplacement de données de masse et des méta-données.
- ◆ Le client SQL*LOADER a été intégré avec les tables externes et fournit la migration automatique des fichiers de contrôle du SQL*LOAD vers les paramètres d'accès aux tables externes.
- ◆ Les clients EXPDP et IMPDP sont des clients légers qui font appel au package DBMS_DATAPUMP pour initialiser et gérer les opérations du DATA Pump. Même s'ils introduisent de nouvelles fonctionnalités, ils restent compatibles avec les clients import et export antérieurs qui sont toujours disponibles.
- ◆ Les applications comme Database Control, la réplication, les tablespaces transportables et les applications utilisateurs bénéficient de cette infrastructure. SQL*Plus peut aussi être utilisé comme un client du package DBMS_DATAPUMP pour des requêtes simples sur l'état des opérations en cours.

30.1 Opérations d'IMPORT et d'EXPORT du DATA Pump

L'import et l'export du DATA Pump sont de nouveaux outils propres à Oracle database 10G.

Ce sont des outils différents des outils d'import et d'export classiques même si les commandes sont similaires.

- ⇒ DATA Pump Export est un outil pour télécharger des données et des métadonnées dans des fichiers du système d'exploitation appelé fichiers de dump.
- ⇒ DATA Pump import est utilisé pour charger des données et métadonnées qui sont stockées dans un fichier de dump vers une base cible.

Le mode de chargement ou de déchargement des outils DATA Pump export et import, est spécifié sur la ligne de commande en utilisant le paramètre approprié. Les divers modes disponibles sont listés ci-dessous, **ce sont les mêmes que ceux des utilitaires d'import et d'export des versions antérieures** :

- ⇒ FULL toute la base (sauf le dictionnaire de données)
- ⇒ SCHEMA tous les objets d'un schéma
- ⇒ TABLE une ou plusieurs tables
- ⇒ TABLESPACE tous les objets contenus dans un tablespace
- ⇒ TABLESPACE Transportable transport d'un tablespace entre é bases

Le cœur de chaque opération DATA Pump est représenté par la table : MASTER TABLE (MT) qui est une table créée dans le schéma de l'utilisateur qui exécute un « job » DATA Pump.

La table MT est construite pendant l'exécution d'un job d'export. Par contre, le chargement de la table MT dans le schéma utilisateur en cours est la première action d'une opération d'import *via* un script, qui est utilisée pour créer la séquence de création de tous les objets importés.



La table `MT` représente l'élément clé pour pouvoir redémarrer l'outil `DATA Pump` dans le cas de l'arrêt planifié ou non planifié du job.

La table `MT` est supprimée quand le `DATA Pump` finit normalement.

30.2 Avantages de l'export et de l'import `DATA Pump`:

- ♦ `DATA Pump` décide automatiquement quelle est la méthode d'accès aux données à utiliser ; cette méthode peut être soit « Direct Pass » soit « Externals tables ».
- ♦ La possibilité de se connecter et se déconnecter à des jobs de longue durée sans affecter le job lui-même, vous permet de superviser ceux-ci à partir des différentes localisations pendant leur temps d'exécution.
- ♦ Les paramètres `EXCLUDE`, `INCLUDE` et `CONTENT` sont utilisés pour sélectionner des objets à des niveaux très fins.
- ♦ Le paramètre `PARALLEL` peut être utilisé pour spécifier le nombre maximal de « Threads » du serveur sur lequel s'exécute le job d'export.
- ♦ Le paramètre `ESTIMATE_ONLY` permet de savoir combien d'espace pourrait être consommé par le job d'export (sans exécuter l'export).
- ♦ Le mode réseau permet d'exporter des objets d'une base distante dans un fichier de dump. Ceci peut être effectué en utilisant un *lien de la base de données* distante vers le système source.
- ♦ Pendant l'import les noms des fichiers de données cibles, les noms des schémas et les noms des tablespaces de la base importée peuvent être changés.

Une fois que le job est déclenché, plusieurs « clients » peuvent se connecter et se déconnecter au job.

Le processus d'arrière plan « *Master Control Process* » (MCP) contrôle l'exécution et la séquence d'un job `DATA Pump` pendant son exécution.

Une fois le job en exécution, la tâche principale du processus d'arrière plan est de desservir les requêtes du « client ».

Si le « client » se déconnecte, le processus d'arrière plan s'arrête.

A la réception d'une requête `START JOB`, le MCP crée un nombre de processus de travail qui dépend de la valeur du paramètre `PARALLEL`. Le nom d'un processus de travail suit le format `DWnn` (il charge et décharge les données).

Si la méthode d'accès aux données est de type `EXTERNAL TABLE PATH` pour le chargement et le déchargement des données, le processus de travail `DWnn` coordonne un nombre parallèle des serveurs d'exécution en fonction du nombre de chargements ou de déchargements définis. Ceci permet le chargement et le déchargement intra-partition.

Remarques

Vous pouvez vous connecter à un job actif afin de l'arrêter, de changer son parallélisme ou bien de superviser son état d'avancement.

Vous pouvez utiliser l'information de la `MT`, dérivée du `JOB_NAME` pour redémarrer un job arrêté ou supprimer toutes les `MT` qui ne sont plus utiles.

Les jobs `DATA Pump` maintiennent une entrée dans la vue `V$SESSION_LONGOPS` sur les performances dynamiques.



Si aucun nom de job n'est spécifié, DATA Pump utilise le schéma du job afin de le générer automatiquement.

Ce nom utilise le format suivant : <USER>_<OPERATION>_<MODE>_%N, nom qui dépend du type d'opération exécutée et de son domaine.

Certains paramètres peuvent entrer en conflit avec d'autres. Par exemple, des valeurs du paramètre TABLES peuvent entrer en conflit avec le paramètre OWNER :

```
OWNER=ORCL  
TABLES=opdef.client
```

Les erreurs n'arrêtent pas l'export, elles sont simplement signalées.

Les possibilités d'export incrémentaux (paramètre INCTYPE) sont obsolètes et conservées pour des raisons de compatibilité.

⇒ Utiliser les fonctionnalités similaires du Recovery Manager



Pour visualiser l'ensemble des paramètres d'export en ligne, utilisez la commande.

Exp help=y

30.3 Le mode interactif du DATA Pump

DATA Pump peut s'exécuter en mode commande (sous dos ou sous unix) en utilisant les commandes expdp ou impdp, avec ou sans fichier de paramètre.

DATA Pump peut s'exécuter également en interactif, ce qui permet de sortir de l'affichage écran du travail en cours qui continue en arrière plan.

⇒ Ainsi en tapant les touches <CTRL+C>, pendant l'exécution du DATA Pump, l'affichage à l'écran s'arrête, mais le travail continue en arrière plan.

30.3.1 Commandes du mode interactif

Dans ce mode il existe un ensemble de commandes qui permettent de gérer les jobs DATA Pump en cours d'exécution.

- ◆ CONTINUE_CLIENT = redémarre l'affichage de l'avancement du job DATA Pump
- ◆ EXIT_CLIENT = quitte l'outil en laissant le travail continuer
- ◆ KILL_JOB = arrête et supprime le job
- ◆ START_JOB = redémarre le job
- ◆ STOP_JOB = arrête le travail sans le supprimer. Sans option, le job termine la tâche en cours avant de s'arrêter. Pour arrêter le job immédiatement, il faut utiliser l'option IMMEDIATE.





La vue DATAPUMP_JOBS permet de visualiser les travaux DATA Pump en cours !

30.4 Méthode d'extraction des données avant et après *data pump*

Dans le chemin conventionnel, l'export avant DATA Pump utilise des `SELECT` pour extraire les données, avec le mécanisme habituel de lecture.

Dans le chemin direct, certaines étapes du mécanisme de lecture sont éliminées, ce qui permet d'améliorer les performances.

Pour pouvoir utiliser le chemin direct, il faut que le jeu de caractères de la session qui réalise l'export soit le même que celui utilisé dans la base (clause `CHARACTER SET` du `CREATE DATABASE`).

La variable d'environnement `NLS_LANG` permet de respecter cette contrainte (par exemple `NLS_LANG = AMERICAN_AMERICA.WE8ISO8859P1`).

Du point de vue des performances, il est conseillé de spécifier un paramètre :

⇒ `RECORDLENGTH` multiple du `DB_BLOCK_SIZE` de la base.

30.4.1 Méthode direct path (chemin direct) du *data pump*

DATA Pump permet 2 méthodes d'accès aux données :

- ⇒ Chemin Direct Path en utilisant l'API direct-path
- ⇒ External tables

DATA Pump sélectionne automatiquement la méthode d'accès la plus adaptée à chaque table.

DATA Pump utilise le chemin « Direct Path » quand la structure de la table le permet et quand il est demandé d'avoir un accès rapide aux données.

Si une des conditions énumérée ci-dessous est remplie, ou encore, si une table contient des colonnes cryptées, ou si les tables chargées sont partitionnées différemment au moment du chargement et du déchargement, le DATA Pump utilisera de préférence les tables externes plutôt que le chemin direct pour déplacer les données.

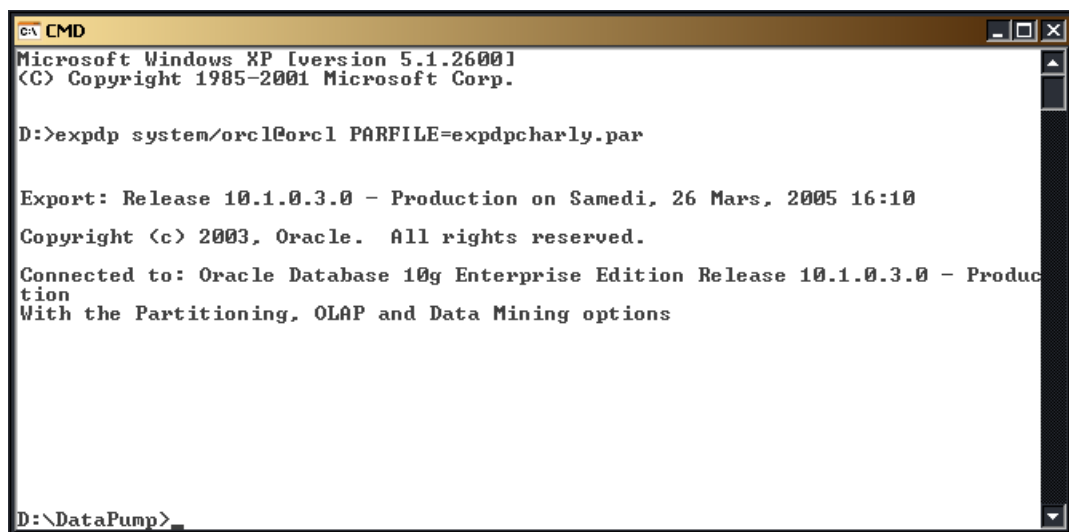
Des données chargées avec une méthode peuvent être déchargées en utilisant l'autre méthode.



30.4.2 Données conduisant un accès utilisant des tables externes :

- ◆ Tables avec contrôle d'accès fin pour le mode INSERT et SELECT
- ◆ Colonnes de type LOB
- ◆ Tables Clusterisées
- ◆ Tables avec triggers actifs
- ◆ Table partitionnées ou avec index globaux sur une seule partition
- ◆ BFILE ou colonnes contenant des tuples opaques (binaire)
- ◆ Contraintes d'intégrité référentielles
- ◆ Colonnes de type VARRAY avec un type opaque encapsulé

L'appel aux outils DATA Pump se fait sous le système d'exploitation, en précisant l'utilisateur de connexion à la base de données et un fichier de paramètres spécifié par le mot clé : PARFILE.



```
C:\> CMD
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:>expdp system/orcl@orcl PARFILE=expdpcharly.par

Export: Release 10.1.0.3.0 - Production on Samedi, 26 Mars, 2005 16:10
Copyright (c) 2003, Oracle. All rights reserved.
Connected to: Oracle Database 10g Enterprise Edition Release 10.1.0.3.0 - Produc
tion
With the Partitioning, OLAP and Data Mining options

D:\DataPump>
```



31 Export/Import Data Pump

31.1 Fichiers supportés par les outils *DATA Pump*

Il y a 3 types de fichiers gérés par les outils *DATA Pump* :

- ⇒ Les fichiers de « dump » qui contiennent les données et les métadonnées à déplacer
- ⇒ Les fichiers de « log » qui tracent les messages associés à chaque opération
- ⇒ Les fichiers « SQL » qui enregistrent le résultat de chaque opération

DATA Pump autorise un accès aux fichiers à travers l'utilisation de chemins d'accès relatifs d'Oracle : les objets *DIRECTORY*. Les chemins absolus ne sont pas supportés pour des raisons de sécurité.

Enchaînement utilisé par les clients *DATA Pump* pour localiser ces fichiers :

- ◆ Les objets *DIRECTORY* par fichier peuvent être spécifiés pour chaque fichier « dump », de « log » et « SQL ». Si ils sont spécifiés, ils sont séparés du nom du fichier par « : »
- ◆ Les clients d'export/import du *DATA Pump* fournissent un paramètre *DIRECTORY* qui spécifie le nom d'un objet *DIRECTORY*. Ces objets *DIRECTORY* décrivent la localisation des fichiers.
- ◆ Une variable d'environnement *DATA_PUMP_DIR* peut être définie pour spécifier le nom de l'objet directory (*variable de chemin par défaut*). Les clients *DATA Pump* vont chercher cette variable d'environnement si aucun objet *DIRECTORY* n'est défini d'une façon explicite.



Il faut avoir les privilèges d'accès aux répertoires pour pouvoir accéder aux fichiers afin de pouvoir exécuter l'opération de chargement ou de déchargement.

Pour l'export, l'accès de type *WRITE* est nécessaire pour tous les fichiers.

Pour l'import, l'accès de type *READ* est nécessaire pour les fichiers de « dump » et l'accès de type *WRITE* est nécessaire pour les fichiers « log » et « SQL ».

```
•      création de la directory dir_charly pour le DATAPump
create or replace DIRECTORY dir_charly
as 'D:\datapump\'
/
```

```
•      APPEL DE LA DIRECTORY DANS LE FICHIER DE PARAMETRES
schemas = CHARLY
directory = dir_charly
dumpfile = ExpdpCharly.dmp
logfile = ExpCharly.log
```



Le paramètre `DUMPFILE` spécifie le nom (répertoires compris) des fichiers de « dump » dans lesquels sont situés les fichiers sur disque.

Le nom du fichier peut contenir la variable de substitution « %U » ce qui signifie que plusieurs fichiers pourront être générés. La variable « %U » sera traduit dans les noms de fichiers résultants, par un numéro à 2 chiffres incrémenté de 1 en 1 commençant à « 01 ».

Si le paramètre `DUMPFILE` n'est pas spécifié, le nom de fichier « expdat.dmp » est utilisé.

Ce sont des fichiers `AUTOEXTENSIBLES`, sauf si le paramètre `FILESIZE` est spécifié.

Chaque fichier aura alors une taille à la valeur de `FILESIZE` et sera non extensible.

S'il n'y a pas assez d'espace et si le format « %U » est défini, un nouveau fichier sera créé automatiquement à la valeur de `FILESIZE` ; sinon le client recevra un message en lui demandant d'ajouter un nouveau fichier.

Si le nom du fichier est généré avec « %U » le nombre de fichiers créés dès le début est égal à la valeur du paramètre `PARALLEL`.

31.2 Paramètres le l'export et de l'import DATA Pump

Ci-dessous sont présentés un ensemble de paramètres utilisés avec DATA Pump. Le type de données manipulées par DATA Pump sont :

- ⇒ ALL : les données et les métadonnées
- ⇒ DATA_ONLY : les données uniquement
- ⇒ METADATA_ONLY : les métadonnées uniquement.

La liste complète des paramètres est expliquée dans la documentation « *Oracle Database Utilities* ».

31.2.1 Paramètres communs

Dans le fichier de paramètres, le caractère # en début de ligne met la ligne en commentaire.

ATTACHE[=schema.]nom_travail]

- Permet d'attacher sa session à un job DATA Pump en cours. Pour attacher un job d'un autre schema, il faut avoir le privilège `EXP_FULL_DATABASE` ou `IMP_FULL_DATABASE`. Si aucun nom n'est spécifié, la session est attachée au travail en cours dans le schéma courant. Si ce paramètre est utilisé, aucun autre paramètre ne peut être spécifié.

JOB_NAME=nom_job

- Permet de donner un nom au job DATA Pump execute. S'il n'est pas précisé le nom du job est `SYS_<opération>_<mode>_<nn>`.

CONTENT={ALL|DATA_ONLY|METADATA_ONLY}

- Permet de définir le contenu de l'export ou de l'import

DIRECTORY=objet_directory

- Permet de définir un répertoire appelé `DIRECTORY` déclaré dans la base de données (par la commande `CREATE DIRECTORY ...`) Les fichiers du DATA Pump iront dans ce répertoire. `DATA_PUMP_DIR` définit la directory par défaut.



DUMPFIL=nom_fichier_dump

- Défini le num du fichier dump en sortie. Il est possible de paralléliser l'export ou l'import.

LOGFILE=nom_fichier_log

- Permet de préciser le nom du fichier de log (fichier journal). Par défaut ces fichiers sont nommés export.log ou import.log. Si le paramètre NOLOGFILE est positionné à « y » alors aucun fichier de log ne sera généré.

NOLOGFILE={y,n}

- Positionné à « y » ce paramètre précise qu'aucun fichier de log ne sera généré.

PARFILE=nom_fichier_paramètres

- Permet de préciser le nom du fichier de paramètre utilisé pour l'export ou l'import. Ce fichier contient les paramètres appliqués lors de l'export ou de l'import. Ce fichier de paramètre doit être présent sur le serveur qui effectue l'export ou l'import.

FULL={y|n}

- Permet de préciser s'il s'agit d'un export ou d'un import complet ou non.

SCHEMAS=nom_schema, ...

- Permet de préciser le nom des schémas à exporter ou à importer

TABLES=[schema.]nom_table[:partition] [, ...]

- Permet de préciser le nom des tables à exporter ou à importer, ainsi que des partition de tables si besoin.

QUERY=[schema.][nom_table :]clause_where

- Permet de filtrer les données à exporter ou à importer

TABLESPACES=nom_tablespace, ...

- Permet de faire un export ou un import de niveau tablespace. Il permet de préciser plusieurs tablespaces.

TRANSPORT_FULL_CHECK={y,n}

- Si ce parameter est positionné à <y>, DATA Pump vérifie les dépendances entre les objets transportés à l'intérieur des tablespaces transportés.

TRANSPORT_TABLESPACES=nom_tablespace, ...

- Permet de faire un export ou un import de niveau transport de tablespace. Il permet de préciser plusieurs tablespaces.

NETWORK_LINK=nom_database_link

- Précise le nom d'un database_link à utiliser pour l'export ou l'import.

EXCLUDE=type_objet[:filtre] [, ...]

- Permet d'exclure des objets pour l'export ou l'import.

INCLUDE=type_objet[:filtre] [, ...]

- Permet d'inclure des objets pour l'export ou l'import.



31.2.2 Paramètres de l'EXPORT DATA Pump

COMPRESSION={ALL|DATA_ONLY|METADATA_ONLY|NONE}

- Active la compression des données ou des « méta-data » du fichier d'export.

ESTIMATE_ONLY={y|n}

- Permet de vérifier l'espace que l'export va occuper sans faire l'export réellement.

31.2.3 Paramètres de l'IMPORT DATA Pump

SQLFILE=nom_fichier_SQL

- Précise le nom du fichier SQL généré au moment de l'import, contenant les ordres DDL correspondant à l'import sans réellement réaliser l'import.

REMAP_SCHEMA=nom_schema_source :nom_schema_cible

- Permet de préciser le nom du schéma qui a été exporté et le nom du schéma dans lequel se fera l'import. Si le schéma n'existe pas il sera créé avec les mêmes privilèges que celui de l'export, par contre le mot de passe devra être modifié avant de pouvoir se connecter.

REMAP_TABLESPACE=nom_tablespace_source :nom_tablespace_cible

- Permet de préciser le nom du tablespace cible lorsque l'on veut changer le tablespace d'origine. Plusieurs paramètres peuvent être spécifiés pour effectuer plusieurs changements de tablespace.

REMAP_DATAFILE=nom_datafile_source :nom_datafile_cible

- Permet de préciser les chemins des fichiers de données des tablespaces cible lorsque l'on veut changer le tablespace d'origine et donc de nom de fichier cible avec des chemins différents. Plusieurs paramètres peuvent être spécifiés pour effectuer plusieurs changements.

TABLE_EXISTS_ACTION=[SKIP|APPEND|TRUNCATE|REPLACE]

- Permet de préciser l'action à effectuer lorsque la table rencontrée au moment de l'import existe déjà :
SKIP = ne rien faire et passer à l'objet suivant (non autorisé si CONTENT=data_only)
APPEND = ajoute les données à la fin du contenu de la table (valeur par défaut si CONTENT=data_only)
TRUNCATE = vide la table avant de charger les données.
REPLACE=supprime la table puis la recrée, avant de charger les données. (non autorisé si CONTENT=data_only).

TRANSPORT_DATAFILES=nom_fichiers, ...

- Permet de préciser l'emplacement des fichiers de données lors d'un transport de tablespace. Les fichiers de données (*datafiles*) doivent être recopiés au préalable.

31.3 Filtrer les données à exporter

Le job DATA Pump peut inclure ou exclure pratiquement tous types d'objets grâce au paramètre EXCLUDE.

Ce paramètre EXCLUDE est spécifié dans un fichier SQL défini par le paramètre SQLFILE.

```
|EXCLUDE=object_type [ : « expression »]
```



Les 3 lignes du fichier de paramètre vont exclure toutes les vues, les packages et les index dont le nom commence par « EMP »

```
EXCLUDE=VIEW  
EXCLUDE=PACKAGE  
EXCLUDE=INDEX : "like 'EMP'"
```

Le paramètre `INCLUDE` permet d'inclure seulement les types d'objets spécifiés et les objets spécifiés pour l'opération.

```
| INCLUDE=object_type [ : « expression »]
```

Le paramètre `CONTENT` permet de sélectionner pour l'opération courante les métadonnées, les données ou les deux :

```
| CONTENT= ALL | METADATA_ONLY | DATA_ONLY
```

Le paramètre `QUERY` fonctionne d'une façon similaire à l'utilitaire d'export antérieur avec 2 améliorations principales :

- ⇒ Il peut être préfixé par un nom de table, pour être appliqué seulement sur cette table.
- ⇒ Il peut être utilisé pour un import.

```
| QUERY= [SCHEMA.] [nom_table :] « QUERY »
```

```
QUERY=charly.employe:"where salaire in (1000,2000)  
And nom_emp like 'Prof%'  
Order by nom_emp"
```



Les paramètres `EXCLUDE` et `INCLUDE` sont mutuellement exclusifs. Ils ne peuvent pas être utilisés si le paramètre `CONTENT=DATA_ONLY`, est spécifié.

Comme la métadonnée de l'objet est stockée sous un format XML dans un fichier de « dump », il sera facile d'appliquer une transformation quand une DDL est définie pendant l'import.

L'import `DATA Pump` permet plusieurs transformations :

- ♦ `REMAP_DATAFILE` est utile pour déplacer des bases de données vers différentes plateformes qui ont des systèmes de gestion de fichiers différentes.
- ♦ `REMAP_TABLESPACE` permet de déplacer des objets d'un tablespace dans un autre.
- ♦ `REMAP_SCHEMA` fournit la fonctionnalité antérieure du `FROMUSER/TOUSER`, ceci afin de changer le propriétaire d'un objet.



- EXEMPLE de Fichier PARFILE pour un import

```
directory = dir_charly  
remap_schema = CHARLY:OPDEF  
dumpFILE = ExpdpCharly.dmp  
LOGfile = Imp.log
```

En utilisant le paramètre `TRANSFORM`, il peut être possible de ne pas générer les clauses de stockage dans la DDL. Ceci est utile si les caractéristiques de stockage de l'instance cible sont très différentes de celles de l'instance source.



Les vues du dictionnaire de données permettant de visualiser la liste des types d'objets pouvant être exporté dans un paramètre `EXCLUDE` ou `INCLUDE` sont :

- `DATABASE_EXPORT_OBJECTS` et
- `SCHEMA_EXPORT_OBJECTS` et
- `TABLE_EXPORT_OBJECTS`

31.4 Exemples d'export et d'import DATA Pump

31.4.1 Estimation de la taille de l'Export

Ce script permet de définir la taille que fera le fichier d'export.

```
> expdp hr/hr DIRECTORY=dpump_dir1 ESTIMATE_ONLY=y TABLES=employees,  
departments, locations LOGFILE=estimate.log
```

31.4.2 Exports Parallélisés

C'est un export de base complet qui a 4 processus de travail parallèles. Les fichiers de dump sont créés dans des répertoires indiqués par les objets `directory` `DATADIR1`, `DATADIR2`, `DATADIR3`, `DATADIR4`.

Chaque fichier a une taille de 2 giga octets et 4 fichiers au moins seront créés.

Le nom du job et de la MT est le nom par défaut `SYSTEM_EXPORT_FULL_01`.

```
Expdp system/bora full = y  
Parallel = 4  
Dumpfile = DATADIR1:full1%U.dat,  
DATADIR2:full2%U.dat,  
DATADIR3:full3%U.dat,  
DATADIR4:full4%U.dat,
```

Exemple 2 : EXPORT Full en Parallel



```
> expdp hr/hr FULL=y DUMPFILE=dpump_dir1:full1%U.dmp, dpump_dir2:full2%U.dmp  
FILESIZE=2G PARALLEL=3 LOGFILE=dpump_dir1:expfull.log JOB_NAME=expfull
```

31.4.3 Import Parallélisé

Cet exemple est un exemple d'import complet du fichier de dump créé avec le 1^{er} exemple.

Le fichier de dump a été envoyé vers un périphérique de stockage réseau spécifié par l'objet directory.

Le paramètre `NET_STORAGE_1.FULL=Y` n'est pas nécessaire car l'import par défaut est celui de l'import complet du fichier de dump. 4 lots parallèles de chargement sont créés.

Le job et la MT ont comme nom par défaut `SYSTEM_IMPORT_FULL_01`.

```
Impdp system/bora  
Directory = NET_STORAGE_1  
Parallel = 4  
Dumpfile = full1%U.dat,  
full2%U.dat,  
full3%U.dat,  
full2%U.dat  
sqlfile = gen08.sql
```

31.4.4 Export de schéma

Des procédures, des packages, des types et des vues dont le nom commence avec `PRODUCT` seront exportés à partir des schémas `CHARLY` et `OPDEF`.

Le fichier de dump « `schema_charly_opdef.dat` » est créé dans le répertoire indiqué dans l'objet directory `USR_DATA`.

Comme l'utilisateur `SYSTEM` détient le rôle `EXPORT_FULL_DATABASE`, il peut spécifier plusieurs schémas.

Les définitions des schémas et les autorisations concernant les privilèges *system* ne sont pas exportés (alors qu'elles devraient l'être normalement) car elles ne sont pas précisées, d'une manière explicite dans la clause `INCLUDE`.

```
Expdp system/bora schemas = CHARLY,OPDEF  
Directory = USR_DATA  
Dumpfile = schema_charly_opdef.dat  
include = function  
include = procedure  
include = package  
include = view:"like 'PRODUCT%'"
```



Exemple 2- export de données déchargées des lignes de Tables

```
|> expdp hr/hr PARFILE=exp.par
```

Le fichier de paramètres contient :

```
DIRECTORY=dpump_dir1  
DUMPFILE=dataonly.dmp  
CONTENT=DATA ONLY  
EXCLUDE=TABLE:"IN ('COUNTRIES', 'LOCATIONS', 'REGIONS')"  
QUERY=employees:"WHERE department_id !=50 ORDER BY employee_id"
```

31.4.5 Import de schéma

Cet exemple d'import montre comment vous pouvez générer un script SQL à partir d'un fichier d'export de dump qui contient toutes les définitions DDL que l'import exécutera en s'appuyant sur les autres valeurs des autres paramètres.

Le SQL ne sera pas exécuté et le système cible restera inchangé.

```
Impdp system/bora directory = USR_DATA  
Dumpfile = schema_CHARLY_OPDEF.dat  
Sqlfile = schema_CHARLY_OPDEF.sql
```

31.5 Remarques et modes opératoires

31.5.1 Export et jeux de caractères

DATA Pump écrit le dump dans le jeu de caractère de la base de données.

Des problèmes peuvent se produire si les jeux de caractères utilisés ne sont pas les mêmes entre la base d'où provient l'export et la base dans laquelle on fait l'import. De même des problèmes peuvent se produire si le jeu de caractère entre la base de données et le jeu de caractère du client ou de l'environnement qui fait l'export ne sont pas compatibles (typiquement, perte des caractères accentués).

Une variable d'environnement `NLS_LANG` correctement positionnée dans l'environnement qui lance l'outil et ouvre la session d'export permet de remédier à ce problème.

L'import peut provoquer une double conversion de jeu de caractères :

- ⇒ Jeu de caractères du fichier d'import (variable d'environnement `NLS_LANG` lors de l'export)
- Jeu de caractères de la session qui effectue l'import (variable d'environnement `NLS_LANG` lors de l'import)
- Jeu de caractères de la base

Historiquement, les outils d'export/import peuvent être utilisés pour réorganiser le stockage de tout ou partie d'une base.

Ce sont toujours les bons outils pour reconstruire une base en changeant la taille de bloc, ou pour changer le jeu de caractère de la base de données.



31.5.2 Remarques sur les dépendances entre les objets

Lors de la récupération de tout ou partie d'un schéma (ou d'une base), si les objets n'existent pas dans la base cible et si le fichier d'export est importé tel quel (sans restructuration du stockage notamment), il n'y a pas de difficulté particulière car Oracle importe les objets dans un ordre « intelligent ».

Le seul problème potentiel concerne l'ordre d'import des vues et des objets stockés vis à vis des dépendances entre les objets ; si un objet est importé avant un autre objet dont il dépend, il est marqué `INVALID` (colonne `STATUS` de la vue `DBA_OBJECTS`) et il doit être recompilé.

Ce problème n'est pas grave, car la recompilation est automatique à la première utilisation et elle ne devrait pas échouer si tous les objets sont présents.

Pour anticiper et éviter tout problème, il est conseillé de vérifier s'il existe des objets invalides et de les recompiler soi-même avec la syntaxe :

```
ALTER { VIEW | PROCEDURE | FUNCTION | PACKAGE | PACKAGE BODY |  
TRIGGER } nom_objet COMPILE
```

La présence de contraintes, d'index ou de triggers déjà existant dans la base cible peut poser plusieurs problèmes, notamment sur les performances et le risque d'avoir des données rejetées.

En ce qui concerne le risque de données rejetées, le problème peut ne pas venir des données proprement dites mais de l'ordre dans lequel l'import est fait : si les données de la table des commandes sont importées avant celles de la table des clients et qu'il y a une contrainte d'intégrité référentielle de la table des commandes vers la table des clients, les commandes risquent d'être rejetées car les clients n'auront pas encore été importés.

La technique classique consiste alors à supprimer ou désactiver les structures gênantes avant l'import et à les réactiver ou les recréer ensuite.

- ⇒ Quels que soient les paramètres, l'import fait une activation des contraintes et des triggers qui étaient actifs au moment de l'export. Si vous aviez désactivé des contraintes et des triggers avant l'import, ils seront réactivés par l'import ; c'est un peu troublant parfois ...
- ⇒ Lorsque l'objectif est de ne transférer que des données d'une base source vers une base cible (la structure est déjà prête dans la base cible), il est préférable de n'exporter que les données sans les contraintes, les index, les triggers, ... Cela permet de limiter les risques de comportements « bizarres » lors de l'import.

31.5.3 Export de niveau tablespace

L'export de niveau tablespace (notion de tablespace transportable) est intéressant pour transférer l'intégralité d'un tablespace d'une base à une autre, sous réserve :

- ♦ Que les bases soient sur la même plate-forme et aient le même jeu de caractères.
- ♦ Pas de tablespace portant le même nom dans la base cible.
- ♦ Que le tablespace soit auto-suffisant (pas de référence vers des objets stockés dans des tablespaces non transportés).



L'export s'effectue dans le jeu de caractères de la session qui effectue l'export, défini par la variable d'environnement `NLS_LANG` (exemple `FRENCH_FRANCE.WE8ISO8859P1`).

Une conversion automatique se produit si ce jeu est différent de celui de la base.

La fonctionnalité de tablespace transportable, introduite en version 8i, est particulièrement intéressante ; elle permet de transporter un tablespace d'une base à une autre, simplement en copiant directement les fichiers de données du tablespace et en le « branchant » sur la base d'arrivée.

Avec la version 9i, il est possible d'utiliser le paramètre `TABLESPACE` pour exporter toutes les tables situées dans un tablespace donné.

Si une table possède une partition dans le tablespace indiqué, elle sera exportée en totalité. Si l'option `indexes=y` est utilisée, les index associés aux tables seront également transportés.

Au moment du transport, le tablespace doit être `READ ONLY`, mais rien n'interdit, ensuite, dans la base source ou dans la base cible, de remettre le tablespace `READ WRITE`.

Le mode opératoire général est le suivant :

S'assurer que le tablespace concerné est `READ ONLY` ; au besoin, le passer `READ ONLY`.

Utiliser l'outil d'export sur la base source (qui contient le tablespace) pour extraire du dictionnaire les informations relatives à ce tablespace :

- ◆ Caractéristiques du tablespace
- ◆ Définition des objets qu'il contient
- ◆ Copier vers la base cible :
 - le fichier d'export contenant les définitions
 - les fichiers de données du tablespace.
- ◆ Utiliser l'outil d'import sur la base cible pour importer dans le dictionnaire les informations relatives au tablespace transporté :
 - Caractéristiques du tablespace
 - Définition des objets qu'il contient

Avec cette technique, à aucun moment, les données proprement dites ne sont ni lues (`SELECT`) par l'export ni insérées (`INSERT`) par l'import : le gain de temps est généralement appréciable.

Il faut *Oracle Enterprise Edition* sur la base source pour faire l'export ; par contre, l'import peut être réalisé sur n'importe quelle gamme Oracle.

31.6 Vues du dictionnaire de données de *DATA Pump*

Vous pouvez utiliser les vues du dictionnaire de données présentées ci-dessous pour obtenir des informations sur les jobs *DATA Pump* :

- `V$SESSION` : liste des sessions utilisateurs
- `V$SESSION_LONGOPS` : performances des sessions, indique l'état d'avancement du job représenté à travers le nombre de méga-bytes de données transférées. L'entrée contient la taille estimée du transfert et elle est mise à jour périodiquement pour refléter la quantité de données transférées.



- **DBA_DATAPUMP_JOBS** : identifie tous les jobs actifs du DATA Pump (quelque soit leur état) d'une instance ou de toutes les instances du RAC (Y sont aussi illustrées toutes les MT qui ne sont pas actuellement associées à un job actif)
- **DBA_DATA_PUMP_SESSIONS** : les sessions utilisateurs correspondant à un job.

```
set linesize 150
col program for A30
col module for A20
select username, program, module, action
from v$session
/
```

USERNAME	PROGRAM	MODULE	ACTION
SYSTEM	sqlplus.exe	SQL*Plus	
SYSTEM	ORACLE.EXE (q000)		
SYSTEM	ORACLE.EXE (DW01)		
SYSMAN	OMS	OEM.SystemPool	
XMLLoader0			
SYSMAN	OMS	OEM.SystemPool	
SYSMAN	OMS	OEM.SystemPool	
DBSNMP	emagent.exe	emagent.exe	
SYSMAN	OMS	OEM.SystemPool	
PingHeartbeatRecorder			
SYSTEM	expdp.exe	expdp.exe	
SYSMAN	OMS	OMS	
SYSMAN	OMS	OEM.BoundedPool	
SYSMAN	OMS	OEM.SystemPool	
DBSNMP	emagent.exe	emagent.exe	
ORACLE.EXE (MMNL)			
ORACLE.EXE (MMON)			
ORACLE.EXE (QMNC)			
SYSMAN	OMS	OMS	
ORACLE.EXE (CJQ0)			
ORACLE.EXE (RECO)			
ORACLE.EXE (SMON)			
ORACLE.EXE (DBW0)			
ORACLE.EXE (MMAN)			
ORACLE.EXE (PMON)			

30 rows selected.



32 SQL*Loader

Parmi les fonctionnalités de SQL*Loader, les suivantes sont particulièrement intéressantes :

- ⇒ Il y a peu de limitation sur le format des données du fichier externe (largeur fixe, avec séparateur, ...).
- ⇒ Plusieurs fichiers externes peuvent être chargés dans la même session.
- ⇒ Plusieurs tables peuvent être chargées dans la même session.
- ⇒ Des critères peuvent être définis pour éliminer certaines données du fichier externe.
- ⇒ Les données peuvent être transformées avec des fonctions SQL pendant le chargement.
- ⇒ Des numéros séquentiels uniques peuvent être générés pour certaines colonnes.

En entrée, SQL*Loader prend un fichier de contrôle qui pilote le chargement (rien à voir avec le fichier de contrôle d'une base) et un ou plusieurs fichiers de données `ASCII` (pas des fichiers de données d'une base Oracle).

En sortie, SQL*Loader alimente la base Oracle et génère un fichier de log, un fichier des rejets (*bad* - données erronées) et un fichier des refus (*discard* - données écartées).

Pour des petits volumes, les données peuvent être directement incluses dans le fichier de contrôle.

Le fichier *discard* contient des enregistrements qui ont été refusés (écartés) par SQL*Loader car ils ne respectaient pas des conditions, des critères, spécifiés dans le fichier de contrôle.

Le fichier *bad* contient des enregistrements qui ont été rejetés soit par SQL*Loader, soit par Oracle:

- ◆ Rejet par SQL*Loader
- ◆ Format de l'enregistrement non valide par rapport à la description du fichier de contrôle
- ◆ Rejet par Oracle
- ◆ Violation d'une contrainte d'intégrité
- ◆ Type de données non valide
- ◆ Rejet par un trigger

Les enregistrements refusés ou rejetés sont écrits tels quels dans les fichiers *bad* et *discard* qui ont donc la même structure que les fichiers de données utilisés en entrée ; après correction éventuelle des enregistrements, les fichiers *bad* et *discard* peuvent être utilisés comme fichiers d'entrée.

Le fichier de log donne énormément d'informations sur le résultat du chargement :

- ◆ Date
- ◆ Nom des fichiers utilisés
- ◆ Paramètres utilisés
- ◆ Tables cibles et mode d'alimentation
- ◆ Conditions éventuelles sur les enregistrements
- ◆ Nombre d'enregistrements chargés
- ◆ Nombre d'enregistrements écartés
- ◆ Nombre d'enregistrements rejetés
- ◆ Messages d'erreurs relatifs aux rejets



SQL*Loader peut effectuer l'import selon deux « chemins », le chemin direct et le chemin conventionnel.

- ⇒ **Chemin direct** = Les données sont chargées en mémoire, formatées dans des blocs qui sont écrits directement dans la base.
- ⇒ **Chemin conventionnel** = Les données sont chargées en mémoire et insérées dans les tables par des ordres `INSERT` classiques. Avec le chemin conventionnel, tous les mécanismes classiques sont appliqués (contraintes, triggers, ...).

Le chemin direct est plus performant mais a les conséquences suivantes :

- ◆ Seules les contraintes `NOT NULL`, `PRIMARY KEY` et `UNIQUE KEY` sont appliquées.
- ◆ Les triggers `INSERT` ne sont pas exécutés.
- ◆ D'autres utilisateurs ne peuvent pas apporter de modifications aux tables.
- ◆ Il faut lancer l'outil dans une fenêtre du système d'exploitation en mettant des paramètres sur la ligne de commande.

Les paramètres peuvent être listés dans un fichier de paramètres dont le nom seul est passé sur la ligne de commande :

```
|C:\>sqlldr parfile=balance.par
```

Il y a deux catégories de paramètres à ne pas confondre :

- ⇒ Les paramètres du fichier de contrôle
- ⇒ Les paramètres de la ligne de commande qui peuvent être listés dans un fichier de paramètres (paramètre de ligne de commande `PARFILE`)
- ⇒ Certains paramètres de la ligne de commande peuvent être inclus dans le fichier de contrôle (paramètre de fichier de contrôle `OPTIONS`) ou sont redondants avec des paramètres du fichier de contrôle !

Les paramètres du fichier de contrôle sont essentiellement destinés à décrire la structure des enregistrements en entrée, les tables cibles et la nature des contrôles/traitements à réaliser sur les enregistrements.

32.1 Fichier de paramètres

Les paramètres de la ligne de commande, ou du fichier de paramètres indiqué sur la ligne de commande, contrôlent le fonctionnement général de l'outil.

Les principaux paramètres de la ligne de commande sont les suivants :

- **BAD** = Nom du fichier *bad* (avec éventuellement un chemin complet). Par défaut égal au nom du fichier de contrôle, mais avec l'extension `.bad`.
- **BINDSIZE** = (65536) , Taille maximum en octets de la *bind array* (« zone de travail »). Contrôle la quantité de données traitée en un seul `INSERT` et la fréquence du `COMMIT` (en corrélation avec le paramètre `ROWS`).
- **CONTROL** = Nom du fichier de contrôle (avec éventuellement un chemin complet).
- **DATA** = Nom du fichier de données à traiter (généralement plutôt indiqué dans le fichier de contrôle). Par défaut égal au nom du fichier de contrôle, mais avec l'extension `.dat`.



- **DIRECT** = (FALSE) TRUE : chemin direct. FALSE : chemin conventionnel.
- **DISCARDFILE** = Nom du fichier discard (avec éventuellement un chemin complet). Par défaut, égal au nom du fichier de contrôle, mais avec l'extension .dsc.
- **DISCARDMAX** = Nombre maximum de rejets autorisés avant l'arrêt du chargement (1 = arrêt au premier). Par défaut, pas d'arrêt.
- **ERRORS** = (50), Nombre d'erreurs d'insertion autorisées avant l'arrêt du chargement (0 = aucune erreur autorisée - mettre un très grand nombre pour ne pas s'arrêter en cas d'erreur). Les enregistrements incriminés sont écrits dans le fichier bad.
- **LOAD** = Nombre maximum d'enregistrements à charger (après SKIP).
- **LOG** = Nom du fichier log (avec éventuellement un chemin complet). Par défaut, égal au nom du fichier de contrôle, mais avec l'extension .log.
- **PARFILE** = Nom du fichier de paramètres (avec éventuellement un chemin complet).
- **READSIZE** = (65536) , Taille en octets du buffer de lecture.
- **ROWS** = (64) , Nombre de lignes par COMMIT. Si ROWS x taille de la ligne est supérieure à BINDISZE, ROWS est automatiquement diminué. Si ROWS x taille de la ligne est inférieur à BINDISZE, ROWS est utilisé tel quel (il n'est pas augmenté).
- **SILENT** = Liste les catégories de message qui ne doivent pas être reportés à l'écran (HEADER, FEEDBACK, ERRORS, DISCARDS, PARTITIONS, ALL).
- **SKIP** = Nombre d'enregistrements à éliminer avant de commencer le chargement (aucun par défaut).
- **USERID** = Paramètres d'ouverture de la session sous la forme :
nom_utilisateur[/mot_de_passe][@nom_service]. Une invite s'affiche pour saisir le mot de passe s'il est non spécifié.
- **SKIP_INDEX_MAINTENANCE** = Mode direct uniquement. YES : les index ne sont pas mis à jour (ils sont marqués UNUSABLE et il faut les reconstruire). NO : les index sont mis à jour.
- **SKIP_UNUSABLE_INDEXES** = YES, autorise le chargement même s'il existe des index préalablement UNUSABLE. NO : n'autorise pas le chargement s'il existe des index préalablement UNUSABLE. Si le cas est rencontré, l'enregistrement est simplement rejeté en chemin conventionnel mais le chargement s'arrête en chemin direct.

Exemples de fichiers de paramètres :

Cas où toutes les informations sont en fait dans le fichier de contrôle.

```
userid=system/manager  
control=balance.ctl
```

Cas où le fichier de contrôle peut s'appliquer sur des fichiers de données de diverses origines (un seul fichier de contrôle et plusieurs fichiers de paramètres).

```
userid=system/manager  
control=balance.ctl  
data=balance_lyon.dat  
log=balance_lyon.log  
bad=balance_lyon.bad  
discardfile=balance_lyon.dsc
```



32.2 Le fichier de contrôle

La syntaxe présentée ci-dessous n'est pas complète, mais les clauses les plus usuelles y sont présentes. Les clauses doivent apparaître dans l'ordre indiqué.

Les lignes de commentaire doivent commencer par deux signes moins (--).

```
[ OPTIONS(liste d'options) ]
LOAD DATA
[ INFILE fichier | *
[ BADFILE fichier ] [ DISCARDFILE fichier ] [ DISCARDMAX valeur ] ]
[ INSERT | APPEND | REPLACE | TRUNCATE ]
INTO TABLE nom
[ INSERT | APPEND | REPLACE | TRUNCATE ]
[ WHEN condition ]
[ FIELDS TERMINATED BY 'x' [ OPTIONALLY ENCLOSED BY 'y' ] ]
[ TRAILING NULLCOLS ]
( colonne [ POSITION(x:y) ] [ type ] [ clause_SQL ],
  ...
)
[ BEGINDATA données ]
```

- LOAD DATA

La clause `INFILE` donne l'emplacement d'un fichier de données à traiter ou est égal au caractère `*` si les données sont dans le fichier de contrôle (clause `BEGINDATA`).

De manière optionnelle, cette clause peut spécifier un fichier bad (option `BADFILE`), un fichier `DISCARD` (option `DISCARDFILE`) et un nombre maximum de rejets autorisés avant l'arrêt du chargement (option `DISCARDMAX`) ; si les paramètres équivalents de la ligne de commande ont été indiqués, ce sont ces derniers qui s'appliquent.

S'il y a plusieurs fichiers à charger en une seule session, plusieurs clauses `INFILE` peuvent être présentes, chaque clause pouvant spécifier ses propres options `BADFILE`, `DISCARDFILE` et `DISCARDMAX`.

Si le fichier de données est indiqué en paramètre de la ligne de commande (`DATA`), la clause est vide (mais il faut laisser le mot clé `LOAD DATA`).

`INSERT | APPEND | REPLACE | TRUNCATE` La clause suivante précise le mode de chargement dans les tables :

- `INSERT` = Ajout, mais uniquement pour une table vide (erreur sinon)
- `APPEND` = Ajout à la table (peut être vide ou non)
- `REPLACE` = Remplace tout le contenu de la table (un ordre `DELETE` est exécuté avant)
- `TRUNCATE` = Remplace tout le contenu de la table (un ordre `TRUNCATE` est exécuté avant)

- INTO TABLE

La clause `INTO TABLE` donne le nom d'une table à charger et décrit comment effectuer le chargement dans cette table. Si plusieurs tables sont chargées à partir d'un même fichier de données, plusieurs clauses `INTO TABLE` sont spécifiées. Pour chaque table, il est possible d'indiquer les options suivantes :

- `INSERT | APPEND | REPLACE | TRUNCATE` : Mode de l'import pour la table
- `WHEN` : Indique une condition sur l'enregistrement pour qu'il soit effectivement chargé dans cette table. La condition peut porter soit sur une colonne de la table cible soit sur un champ de l'enregistrement source défini par la position de son caractère de début et la position de son caractère de fin sous la forme « début:fin ».
- `FIELDS TERMINATED BY 'x' [OPTIONALLY ENCLOSED BY 'y']` : Pour des enregistrements de longueur variable (avec séparateur), indique comment sont délimités les champs avec :



TERMINATED BY 'x' : caractère séparateur des enregistrements (une virgule,...)

OPTIONALLY ENCLOSED BY 'y' : caractère pouvant entourer les enregistrements (apostrophes, ...)

TRAILING NULLCOLS : Les colonnes non présentes à la fin de l'enregistrement sont mises à NULL (si l'option est absente et que des colonnes vides existent à la fin, l'enregistrement est rejeté).

• Colonne [POSITION(x:y)] [type] [clause_SQL] Liste des colonnes à alimenter dans la table avec :

COLONNE = le nom de la colonne cible

POSITION(x:y) : position du champ correspondant dans l'enregistrement source (cas d'un enregistrement de longueur fixe) ; Pour des enregistrements avec séparateur, la correspondance colonne/enregistrement est en lien avec la position (1^{ère} colonne de la liste = 1^{er} enregistrement, ...)

TYPE : type de données (en cas d'ambiguïté)

CLAUSE_SQL : clause SQL à appliquer

Pour référencer une colonne x dans la clause SQL, utiliser la syntaxe :

x (caractère deux points devant le nom de la colonne).

Il existe d'autres options sur la spécification des colonnes.

BEGINDATA

La clause BEGINDATA marque le début des données si celles-ci sont incluses dans le fichier de contrôle (INFILE *).

32.3 Exemples de chargements

Les différents exemples sont présentés avec des données incluses (INFILE * + BEGINDATA) pour mieux visualiser la correspondance entre les données et les paramètres du fichier de contrôle.

Ces exemples sont très simples à adapter au chargement d'un fichier externe :

- ⇒ Copier les données, sans le BEGINDATA, dans un fichier.
- ⇒ Mettre le nom du fichier dans la clause INFILE (à la place du caractère *).
- ⇒ Supprimer la clause BEGINDATA.

Pour ces exemples, nous supposons l'existence des tables suivantes :

```
• tables des Employe_BIS
CREATE TABLE Employe_BIS (
Code NUMBER(6),
nom VARCHAR2(40),
prenom VARCHAR2(40),
sexe CHAR(1),
date_naissance DATE,
adresse VARCHAR2(150)
)
/

• tables des Employe_BIS masculins (pas de colonne sexe)
CREATE TABLE Employe_BIS_M (
Code NUMBER(6),
nom VARCHAR2(40),
prenom VARCHAR2(40),
date_naissance DATE,
adresse VARCHAR2(150)
)
/
```




```
• tables des Employe_BIS féminins(pas de colonne sexe  
• ni de colonne date_naissance)  
CREATE TABLE Employe_BIS_F (  
code NUMBER(6),  
nom VARCHAR2(40),  
prenom VARCHAR2(40),  
adresse VARCHAR2(80)  
)  
/  
  
• séquence utilisée pour alimenter les numéros d'employé  
CREATE SEQUENCE SEQ_Employe_BIS  
/  
/
```

32.3.1 Exemples de fichiers de contrôle : Longueur variable enregistrements

```
LOAD DATA  
INFILE *  
INTO TABLE Employe_BIS  
APPEND  
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'  
TRAILING NULLCOLS  
(  
prenom ,  
nom ,  
sexe ,  
date_naissance « TO_DATE(:date_naissance,'YYYYMMDD') »,  
adresse,  
code "SEQ_Employe_BIS.NEXTVAL"  
)  
BEGINDATA  
Pierre,DUPOND,M,19660826, "2 rue de Matignon 78711 Mantes la Ville"  
Marise,LEROY,F, "125 rue de Champion 75000 Paris"  
Jean, »PEYRAC (de) »,M,19631204, "6 rue du Faubourg St Antoine 75000 Paris"
```

Pour cet exemple, les enregistrements ont une longueur variable, avec séparateur.

Spécifications des données en entrée :

- ♦ Les champs sont délimités par une virgule. Clause `TERMINATED BY ','`
- ♦ Ils peuvent éventuellement être entourés de guillemets (c'est le cas du nom dans la troisième ligne). Clause `OPTIONALLY ENCLOSED BY '"'`
- ♦ Ils peuvent être manquants en fin de ligne (pas de date de naissance sur la deuxième ligne) ; dans ce cas mettre un `NULL`. Clause `TRAILING NULLCOLS`
- ♦ La date de naissance est fournie sous forme de chaîne au format `YYYYMMDD` mais doit être stockée dans une colonne de type `DATE`. Clause SQL `« TO_DATE(:date_naissance,'YYYYMMDD') »` ; Noter le caractère deux points devant le nom de la colonne `date_naissance` pour la référencer dans le calcul.
- ♦ Le numéro d'adhérent n'est pas fourni ; il doit être calculé à l'aide d'une séquence. Clause SQL `"SEQ_Employe_BIS.NEXTVAL"`



Avec des enregistrements de longueur fixe, la correspondance entre les champs et les colonnes sont définies par la clause `POSITION` ; les colonnes qui ne sont pas alimentées par un champ de l'enregistrement peuvent être spécifiées n'importe où dans la liste des colonnes.

Dans cet exemple, les `EMPLOYE_BIS` de sexe féminin ne doivent pas être chargés.

Une clause `WHEN` (`sexe = 'M'`) est ajoutée pour spécifier les enregistrements à conserver.

32.3.2 Exemples de fichiers de contrôle : Longueur fixe avec élimination d'enregistrements

```
LOAD DATA
INFILE *
INTO TABLE Employe_BIS
APPEND
WHEN (sexe = 'M')
TRAILING NULLCOLS
(
code      "SEQ_Employe_BIS.NEXTVAL",
prenom    POSITION(01:10),
nom       POSITION(11:22),
sexe      POSITION(23:23),
date_naissance POSITION(24:31) « TO_DATE(:date_naissance,'YYYYMMDD') »
)
BEGINDATA
Pierre    DUPOND      M19660826      "2 rue de Matignon 78711 Mantes la Ville"
Marise    LEROY       F              "125 rue de Champion 75000 Paris"
Jean      PEYRAC (de) M19631204  "6 rue du Faubourg St Antoine 75000 Paris"
```

32.3.3 Chargement dans deux tables

```
LOAD DATA
INFILE *
INTO TABLE Employe_BIS_M
APPEND
WHEN ((23) = 'M')
TRAILING NULLCOLS
(
code      "SEQ_Employe_BIS.NEXTVAL",
prenom    POSITION(01:10),
nom       POSITION(11:22),
date_naissance POSITION(24:31) « TO_DATE(:date_naissance,'YYYYMMDD') »
)
INTO TABLE Employe_BIS_F
APPEND
WHEN ((23) = 'F')
TRAILING NULLCOLS
(
code      "SEQ_Employe_BIS.NEXTVAL",
prenom    POSITION(01:10),
nom       POSITION(11:22)
)
BEGINDATA
Pierre    DUPOND      M19660826      "2 rue de Matignon 78711 Mantes la Ville"
Marise    LEROY       F              "125 rue de Champion 75000 Paris"
Jean      PEYRAC (de) M19631204  "6 rue du Faubourg St Antoine 75000 Paris"
```

Pour cet exemple, les enregistrements ont une longueur fixe mais les `EMPLOYE_BIS` doivent être répartis entre deux tables en fonction de leur sexe.



Les deux tables n'ont pas de colonne sexe et la table des EMPLOYE_BIS de sexe féminin, pas de colonne date_naissance non plus.

Le fichier de contrôle utilise deux clauses INTO TABLE pour spécifier comment alimenter les deux tables.

Dans chaque clause INTO TABLE, une clause WHEN ((23) = 'X') permet d'indiquer si l'enregistrement courant doit être chargé dans la table ou non.

Comme les tables n'ont pas de colonne sexe, le seul moyen de désigner le champ correspondant est d'utiliser une notation par position du type (début:fin) ; le champ sexe commence (et finit) au 23^{ème} caractère, soit (23:23) qui peut être abrégé en (23).

Chaque clause INTO TABLE a sa propre liste de colonnes.

32.3.4 Chargement dans deux tables avec utilisation d'une colonne FILLER

```
LOAD DATA
INFILE *
INTO TABLE Employe_BIS_M
APPEND
WHEN (sexe = 'M')
TRAILING NULLCOLS
(
code                                "SEQ_Employe_BIS.NEXTVAL",
prenom                             POSITION(01:10),
nom                                POSITION(11:22),
sexe                                FILLER          POSITION(23:23),
date_naissance                     POSITION(24:31)
« TO_DATE(:date_naissance,'YYYYMMDD') »
)
INTO TABLE Employe_BIS_F
APPEND
WHEN (sexe = 'F')
TRAILING NULLCOLS
(
code                                "SEQ_Employe_BIS.NEXTVAL",
prenom                             POSITION(01:10),
nom                                POSITION(11:22),
sexe                                FILLER          POSITION(23:23)
)
BEGINDATA
Pierre    DUPOND                M19660826    "2 rue de Matignon 78711 Mantes la Ville"
Marise    LEROY                  F        "125 rue de Champion 75000 Paris"
Jean      'PEYRAC (de)'          M19631204    "6 rue du Faubourg St Antoine 75000 Paris"
```

Cet exemple est une variante de l'exemple précédent dans lequel l'enregistrement correspondant au sexe est matérialisé et nommé pour faciliter sa manipulation, bien qu'il n'alimente pas une colonne des tables.

Colonne nommée dans la liste des colonnes avec la propriété FILLER (« remplissage »).

Cette « colonne » peut ensuite être manipulée comme si c'était une colonne de la table (utilisée dans une clause WHEN, dans une clause SQL) mais elle n'est pas chargée dans la table.



32.4 Chargement de données LOB

Cet exemple provient de la documentation Oracle « *b10825.pdf* ».

Chargement de données à partir de fichiers contenant les types LOB.

```
Control File Contents
LOAD DATA
INFILE 'sample.dat'
INTO TABLE person_table
FIELDS TERMINATED BY ','
(name CHAR(20),
1 ext_fname FILLER CHAR(40),
2 "RESUME" LOBFILE(ext_fname) TERMINATED BY EOF)
```

Datafile (sample.dat)

```
Johnny Quest,jqresume.txt,
Speed Racer,'/private/sracer/srresume.txt',
Secondary Datafile (jqresume.txt)
Johnny Quest
500 Oracle Parkway
...
```

Deuxième Datafile (srresume.txt)

```
Speed Racer
400 Oracle Parkway
...
```

32.5 Chargement de formats XML

Il est possible de manipuler des formats XML en enregistrant un schéma xsl dans oracle comme type de donnée(par exemple personne.xsd).

IL est ensuite possible charger des données xml respectant Ce format dans la table.

Ce chargement peut se faire en utilisant les types suivant :

- ♦ Chargement de types XML à partir de types Primaires
- ♦ Chargement de types XML à partir de types BFILE
- ♦ Chargement de types XML à partir de types LOBFILES

La documentation ci-dessous vous fournit un exemple d'utilisation.

[HTTP://DOWNLOAD.Oracle.COM/DOCS/CD/B19306_01/APPDEV.102/B14259/XDB25LOA.HTM](http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14259/xdb25loa.htm)



Example 29-1 Loading Very Large XML Documents Into Oracle Database Using SQL*Loader

Cet exemple utilise le fichier de control `load_data.ctl` pour charger des données de type XML dans la table `foo`. Le code enregistre le format XMLTYPE créé nommé `person.xsd` dans la table `foo`.

Cet exemple est extrait de la documentation oracle *xdb2510a*.

```
CREATE TYPE person_t AS OBJECT(name VARCHAR2(100), city VARCHAR2(100));/
BEGIN
  • Delete schema if it already exists (else error)
  DBMS_XMLSCHEMA.deleteSchema('http://www.oracle.com/person.xsd', 4);
END;/
BEGIN
  DBMS_XMLSCHEMA.registerSchema('http://www.oracle.com/person.xsd',
    <schema xmlns="http://www.w3.org/2001/XMLSchema" ||
    , xmlns:per="http://www.oracle.com/person.xsd" ||
    , xmlns:xdb="http://xmlns.oracle.com/xdb" ||
    , elementFormDefault="qualified" ||
    , targetNamespace="http://www.oracle.com/person.xsd"> ||
    <element name="person" type="per:persontype" ||
    ` xdb:SQLType= »PERSON_T »/>` ||
    <complexType name="persontype" xdb:SQLType="PERSON_T"> ||
    <sequence> ||
    <element name="name" type="string" xdb:SQLName="NAME" ||
    ` xdb:SQLType="VARCHAR2"/>` ||
    <element name="city" type="string" xdb:SQLName="CITY" ||
    ` xdb:SQLType="VARCHAR2"/>` ||
    </sequence> ||
    </complexType> ||
    </schema>,
    TRUE,
    FALSE,
    FALSE);
END;/

CREATE TABLE foo OF XMLType
XMLSCHEMA „http://www.oracle.com/person.xsd“ ELEMENT „person“;
```

Here is the content of the control file, `load_data.ctl`, for loading XMLType data using the registered XML schema, `person.xsd`:

```
LOAD DATA
INFILE *
INTO TABLE foo TRUNCATE
XMLType(xmldata)
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(
xmldata
)
BEGINDATA
<person xmlns="http://www.oracle.com/person.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/person.xsd
http://www.oracle.com/person.xsd"> <name> xyz name 2</name> </person>
```

Here is the SQL*Loader command for loading the XML data into Oracle Database:

`sqlldr [username]/[password] load_data.ctl (optional: direct=y)`

In `load_data.ctl`, the data is present in the control file itself, and a record spanned only one line (it is split over several lines here, for printing purposes).

In the following example, the data is present in a separate file, `person.dat`, from the control file, `lod2.ctl`. File `person.dat` contains more than one row, and each row spans more than one line. Here is the control file, `lod2.ctl`:



```
LOAD DATA
INFILE *
INTO TABLE foo TRUNCATE
XMLType(xmldata)
FIELDS(fill filler CHAR(1),
xmldata LOBFILE (CONSTANT person.dat)
TERMINATED BY '<!-- end of record -->')
BEGINDATA
0
0
0
```

The three zeroes (0) after BEGINDATA indicate that three records are present in the data file, person.dat. Each record is terminated by <!-- end of record -->. The contents of person.dat are as follows:

```
<person xmlns="http://www.oracle.com/person.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/person.xsd
http://www.oracle.com/person.xsd">
<name>xyz name 2</name>
</person>
<!-- end of record -->
<person xmlns="http://www.oracle.com/person.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/person.xsd
http://www.oracle.com/person.xsd">
<name> xyz name 2</name>
</person>
<!-- end of record -->
<person xmlns="http://www.oracle.com/person.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.oracle.com/person.xsd
http://www.oracle.com/person.xsd">
<name>xyz name 2</name>
</person>
<!-- end of record -->
```

Here is the SQL*Loader command for loading the XML data into Oracle Database:

sqlldr [username]/[password] lod2.ctl (optional: direct=y)



33 Stratégie de Sauvegards et Restaurations

La principale responsabilité du DBA est de prendre les mesures nécessaires pour assurer la sécurité et la disponibilité des données.

⇒ Il doit restituer les données en cas d'incident matériel ou d'erreur de manipulation.

Cette sécurité est assurée par :

- ♦ La mise en œuvre d'une protection des fichiers sensibles de la base
 - Fichiers de contrôle
 - Fichiers de Redo Log
- ♦ La mise en place d'une stratégie de sauvegarde/restauration
 - Adaptée aux contraintes de l'entreprise
 - Et qui aura été complètement testée et documentée



Le DBA doit faire des sauvegards régulières de la base de données.



Sauvegards Physiques ET



Sauvegards logiques

Les questions à se poser pour définir la stratégie sont les suivantes :

- ⇒ Est-il acceptable de perdre des données ?
 - si oui sur quelle période ?
- ⇒ Est-il possible d'arrêter périodiquement la base ?
 - si oui quand et combien de temps ?
 - Est-il possible de faire une sauvegarde complète de la base pendant l'arrêt de celle-ci ?

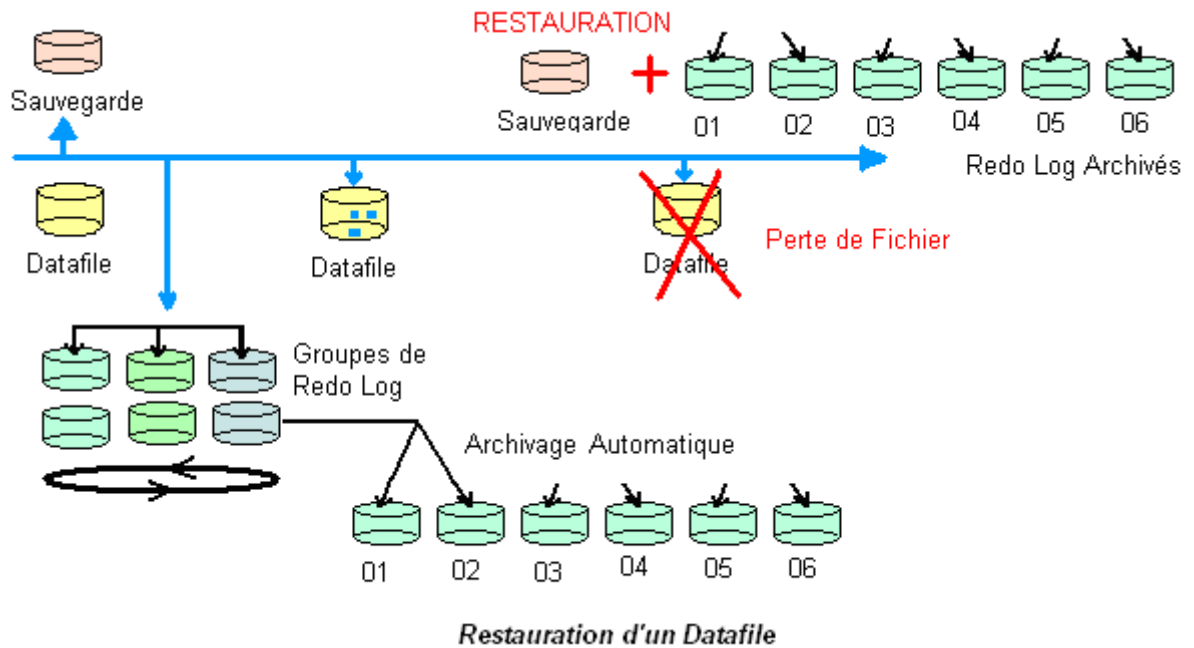
S'il n'est pas possible d'arrêter la base ou de faire une sauvegarde complète, il sera toujours possible de faire des sauvegards partielles.

La sauvegarde d'une base de petits volumes, contenant des données stratégiques ou de volumes importants est différente. Par exemple, pour certaines bases de données, il sera préférable de recharger certaines tables plutôt que de les restaurer (gain de temps pour de gros volumes).

Ne pas oublier que le tablespace est l'unité de sauvegarde.

Il faudra penser au stockage des tables dans des tablespaces dédiés (les grosses tables ou les tables stratégiques) ou mutualisés (les petites tables de référence)





La restauration du fichier de données consiste à prendre la dernière sauvegarde du fichier de données, de remplacer par cette sauvegarde le fichier manquant ou endommagé et à appliquer sur cette sauvegarde les fichiers de Redo Log archivés.

Ceci afin de ramener le fichier de données dans l'état où il se trouvait juste avant l'incident.

33.1 Les modes NOARCHIVELOG et ARCHIVELOG

En fonction des possibilités d'arrêt et des pertes de données (acceptables), on choisira de mettre la base de données en mode ARCHIVELOG ou non.

En cas de passage en mode ARCHIVELOG, il faut étudier :

- ⇒ Où mettre les archives (sur disque ou sur bande) ?
- ⇒ Combien de jeux de sauvegarde conserve-t-on (épuration des sauvegardes et des archives) ?

Oracle écrit en général de manière cyclique dans les groupes de Redo Log en ligne qui forment le journal de reprise.

Oracle passe au fichier suivant lorsqu'il en a rempli un.

Lorsque le dernier fichier affecté au journal est plein, le processus d'arrière-plan LGWR (Log Writer) commence alors à écraser le contenu des membres du premier groupe de Redo Log.

Lorsqu'Oracle est exécuté en mode ARCHIVELOG, le processus ARCH, effectue une copie d'un membre de Redo Log après que le processus LGWR ait fini d'écrire.



33.1.1 Le mode NOARCHIVELOG

Un fichier de Redo Log peut être réutilisé immédiatement après qu'un point de synchronisation (*checkpoint*) ait eu lieu.

Après que le contenu d'un fichier de Redo Log ait été écrasé par une nouvelle écriture, les données à restaurer sont perdues.

Affichage du mode courant :

```
Select log_mode from v$database ;  
LOG_MODE  
NOARCHIVELOG
```

33.1.2 Le mode ARCHIVELOG

La synchronisation des fichiers de données est basée sur le numéro de séquence du fichier de Redo Log (sauf si le tablespace est en READ ONLY).

```
• rechercher le dernier N° de SCN (transaction=99999999).  
-- qui correspond à la colonne FIRST_CHANGE#.  
SQL> select * from v$log_history;
```

RECID	STAMP	THREAD#	SEQUENCE#	FIRST_CHANGE#	FIRST_TI	NEXT_CHANGE#
1	568375848	1	1	2711661	07/09/05	2714944
2	568375866	1	2	2714944	07/09/05	2715550
3	568375884	1	3	2715550	07/09/05	2716153
4	568375902	1	4	2716153	07/09/05	2716755
5	568375919	1	5	2716755	07/09/05	2717359
6	568375937	1	6	2717359	07/09/05	2717961
7	568375954	1	7	2717961	07/09/05	2718563
8	568375971	1	8	2718563	07/09/05	2719168
9	568375989	1	9	2719168	07/09/05	2719770
10	568376006	1	10	2719770	07/09/05	2720373
11	568376024	1	11	2720373	07/09/05	2720976

En cas de restauration, Oracle a besoin de tous les fichiers de Redo Log à partir de celui portant le numéro de séquence inscrit dans le fichier de données au moment de la sauvegarde. Autrement la restauration ne sera pas possible.



Si un fichier d'archive est perdu, Oracle ne pourra pas restaurer la base de données.



33.1.3 Mettre la base en mode ARCHIVELOG

Ce mode permet de garantir qu'un groupe de fichiers de Redo Log non archivés ne sera pas écrasé par LGWR. En version 9i, mettre la base en ARCHIVELOG ne démarrait pas automatiquement le processus ARCH. Il fallait le faire en positionnant le paramètre `LOG_ARCHIVE_START` ⑨ à TRUE.

⇒ En cas d'oubli de positionnement de ce paramètre, cela provoquait un crash de l'instance.

Pour débloquer la situation il fallait lancer la commande `ARCHIVE LOG ALL`, puis monter la base (`STARTUP MOUNT`) et lancer la commande `ARCHIVE LOG ALL`.

Démarrer le processus ARCH

⇒ Se charge de l'archivage proprement dit

⑩ A partir de la version 10g, placer la base en mode ARCHIVELOG démarre automatiquement les processus ARCH0 et ARCH1 lors de l'ouverture de la base de données.

Lors de l'activation de l'archivage, il est primordial de bien vérifier que tout fonctionne bien et que des archives sont bien générées.

33.1.4 Les paramètres du processus ARCH

Les différents paramètres relatifs au mode ARCHIVELOG sont présentés ci-dessous.

- `LOG_ARCHIVE_START` ⑨ (en version 9i)

Démarre (valeur TRUE) ou non (valeur FALSE, par défaut) le processus ARCH

- `LOG_ARCHIVE_DEST_i` = 'LOCATION=chemin_local'

- • (i de 1 à 10)

Destination de l'archivage pour une ENTERPRISE édition (au moins une obligatoire)

Syntaxe simplifiée pour une destination locale (au moins une obligatoire)

Exemple :

`LOG_ARCHIVE_DEST_1 = 'LOCATION=d:\oracle\admin\BORA\arch'` [Pas de blanc pour 'location=']

Le répertoire spécifié doit exister ; il n'est pas créé par Oracle.

ATTENTION, l'utilisation des derniers paramètres est destinée au DATA GUARD.

- `LOG_ARCHIVE_FORMAT`

Format souhaité pour le nom des archives (les 3 sont obligatoires)

%s ou %S : numéro de séquence du fichier de Redo Log

%t ou %T : numéro d'instance

%r ou %R : identifiant de remise à zéro des fichiers de journalisation

Exemple :

`LOG_ARCHIVE_FORMAT = « 'Redo%S_%R%T.arch' »`



- **ARCHIVE_LAG_TARGET**

Durée maximale en seconde qui doit séparer 2 archivages.

Une valeur nulle désactive la fonctionnalité (valeur par défaut).

Valeur autorisée : entre 60 (1 minute) et 7200 (2 heures) ; permet de forcer l'archivage de façon périodique et de garantir une périodicité d'archivage stable.

Exemple :


Archive_lag_target = 1800 #30 minutes

33.2 Passer la base en mode ARCHIVELOG

Avant de passer la base en mode ARCHIVELOG, arrêtez celle-ci et effectuez une sauvegarde à froid.

Ne pas oublier de sauvegarder le fichier SPFILE.

En version 11g passer une base de données ORACLE en mode ARCHIVELOG consiste à monter la base de données et à changer son état. En effet, les destinations d'archive et les noms des archives générées sont paramétrés par défaut.

MODE OPERATOIRE	
	↗ Arrêter la base de données SHUTDOWN IMMEDIATE
	↗ Monter la base STARTUP MOUNT
	↗ Passer la base en ARCHIVELOG ALTER DATABASE ARCHIVELOG ;
	↗ Ouvrir la base ALTER DATABASE OPEN

Après avoir passé la base en mode ARCHIVELOG, vérifiez que les archives sont générées correctement et apparaissent à l'emplacement demandé, puis **arrêtez la base de données et faites une première sauvegarde.**

```
• passer la base en mode ARCHIVELOG
Create pfile from spfile ;
alter system set
LOG_ARCHIVE_DEST_1 = 'location=f:\oracle\oradata\BORA\arch\'
scope=SPFILE ;

ALTER SYSTEM SET LOG_ARCHIVE_FORMAT='arch%S_%R%T.arc'
scope=SPFILE ;

Shutdown immediate;
Startup mount;
Alter database archivelog;
Alter database open;
```



33.3 Administrer le processus *ARCH*

La clause `ARCHIVE LOG` de l'ordre SQL : `ALTER SYSTEM` permet d'administrer le processus `ARCH` après démarrage de la base :

- ⇒ Arrêter/Démarrer le processus `ARCH`
- ⇒ Archiver manuellement un groupe de fichiers de Redo Log

```
ALTER SYSTEM ARCHIVE LOG  
STOP  
| START [ TO 'destination' ]  
| GROUP numero [ TO 'destination' ]  
;
```

- Arrêter le processus `ARCH`
`ALTER SYSTEM ARCHIVE LOG STOP;`
- Démarrer le processus `ARCH`
`ALTER SYSTEM ARCHIVE LOG START;`
- Archiver le groupe 2 vers une autre destination
`ALTER SYSTEM ARCHIVE LOG GROUP 2 TO 'd:\temp' ;`



L'ordre SQL : `ALTER SYSTEM` ne dure que pendant l'instance en cours et ne dure pas après un arrêt de la base.

La commande `ARCHIVE LOG LIST` permet d'afficher des informations sur l'archivage :

```
Connect / as sysdba  
Connected.  
Archive log list  
Database log mode           Archive Mode  
Automatic archival          Enabled  
Archive destination         h:\oracle\oradata\ORCL\archive  
Oldest online log sequence  6355  
Next log sequence to archive 6358  
Current log sequence        6358
```

- **Database log mode** = mode de fonctionnement de la base
- **Automatic archival** = état du processus `ARCH` (enabled s'il est lancé, disabled s'il est arrêté)
- **Archive destination** = destination des archives
- **Oldest online log sequence** = plus ancien numéro de séquence des fichiers de Redo Log en ligne
- **Next log séquence to archive** = prochain numéro de séquence des fichiers de Redo Log à archiver (ligne absente si la base est en `NOARCHIVELOG`)
- **Current log séquence** = numéro de séquence des fichiers de Redo Log courants

33.3.1 Forcer l'archivage de façon périodique

En règle générale, il est conseillé d'avoir des basculements de fichiers de Redo Log (et donc des archivages en mode `ARCHIVELOG`) toutes les 20 à 30 minutes.



⇒ Plus l'activité est importante, plus il y a de basculement et génération d'archivages !

Pour garantir une périodicité stable, il est possible de spécifier une valeur dans le paramètre `ARCHIVE_LAG_TARGET` :

- ◆ Indique en secondes la durée maximale qui doit séparer deux archivages
- ◆ Une valeur `nulle` désactive la fonctionnalité (valeur par défaut)
- ◆ Valeurs autorisées : entre 60 (une minute) et 7200 (2 heures)
- ◆ Le paramètre est dynamique



34 Sauvegardes

Pour connaître l'emplacement et le nom des fichiers de la base, il existe une astuce qui consiste à générer la trace du fichier de contrôle.

Cette trace vous permettra de connaître l'emplacement et le nom des fichiers de la base de données mais aussi de recréer le fichier de contrôle en cas de celui-ci ou en cas de restauration partielle de la base de données.

```
SQL> show parameter user_dump_dest
NAME                                TYPE        VALUE
-----
diagnostic_dest                     string      D:\Oracle\admin\COLLEGE\trace
```

On possède un script de recréation qui a été réalisé avec la commande :

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE ;
```

```
*** 2004-06-14 10:46:04.000
# The following are current System-scope REDO Log Archival related
# parameters and can be included in the database initialization file.
#
# Below are two sets of SQL statements, each of which creates a new
# control file and uses it to open the database. The first set opens
# the database with the NORESETLOGS option and should be used only if
# the current versions of all online logs are available. The second
# set opens the database with the RESETLOGS option and should be used
# if online logs are unavailable.
# The appropriate set of statements can be copied from the trace into
# a script file, edited as necessary, and executed when there is a
# need to re-create the control file.
#
#      Set #1. NORESETLOGS case
#
# The following commands will create a new control file and use it
# to open the database.
# Data used by the recovery manager will be lost. Additional logs may
# be required for media recovery of offline data files. Use this
# only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "OPTIMUM" NORESETLOGS NOARCHIVELOG
•      SET STANDBY TO MAXIMIZE PERFORMANCE
MAXLOGFILES 32
MAXLOGMEMBERS 5
MAXDATAFILES 128
MAXINSTANCES 16
MAXLOGHISTORY 1815
LOGFILE
GROUP 1 (
'D:\ORACLE\ORADATA\OPTIMUM\REDO01A.LOG',
'D:\ORACLE\ORADATA\OPTIMUM\REDO01B.LOG'
) SIZE 10M,
GROUP 2 (
'D:\ORACLE\ORADATA\OPTIMUM\REDO02A.LOG',
'D:\ORACLE\ORADATA\OPTIMUM\REDO02B.LOG'
) SIZE 10M,
GROUP 3 (
'D:\ORACLE\ORADATA\OPTIMUM\REDO03A.LOG',
'D:\ORACLE\ORADATA\OPTIMUM\REDO03B.LOG'
) SIZE 10M
•      STANDBY LOGFILE
DATAFILE
'D:\ORACLE\ORADATA\OPTIMUM\SYSTEM01.DBF',
'D:\ORACLE\ORADATA\OPTIMUM\UNDOTBS01.DBF',
'D:\ORACLE\ORADATA\OPTIMUM\ELEVE01.DBF',
'D:\ORACLE\ORADATA\OPTIMUM\INDX01.DBF',
```



```
'D:\ORACLE\ORADATA\OPTIMUM\OPDEF01.DBF'  
CHARACTER SET WE8ISO8859P15  
;  
# Recovery is required if any of the datafiles are restored backups,  
# or if the last shutdown was not normal or immediate.  
RECOVER DATABASE  
# Database can now be opened normally.  
ALTER DATABASE OPEN;  
# Commands to add tempfiles to temporary tablespaces.  
# Online tempfiles have complete space information.  
# Other tempfiles may require adjustment.  
ALTER TABLESPACE TEMP ADD TEMPFILE 'D:\ORACLE9\ORADATA\OPTIMUM\TEMP01.DBF'  
SIZE 10485760 REUSE AUTOEXTEND OFF;  
# End of tempfile additions.  
#
```

Le fichier trace « .TRC » qui est créé avec cette commande ne peut pas être lancé directement.

Il faut exclure toutes les lignes de début de fichier qui ne sont pas des commentaires et vérifier la syntaxe de la commande `STARTUP`.

Cette commande ne précise pas de fichier de paramètres. Il faut donc rajouter à la fin de la commande `STARTUP` le nom et le chemin du fichier de paramètres.

Pour se connecter sous l'instance où l'on souhaite ouvrir la base et lancer le script de création du fichier de contrôle, la base de données doit être à l'état `NOMOUNT`.

Après exécution, la base est ouverte et le fichier de contrôle récupéré.

34.1 Sauvegarde base arrêtée

Les sauvegardes hors ligne se produisent lorsque la base a été arrêtée proprement.

Pas de panne instance ou `shutdown abort` !

Les fichiers suivants doivent être sauvegardés :

- ♦ Fichiers de contrôle
- ♦ Fichiers de données
- ♦ Fichiers de Redo Log en ligne
- ♦ Fichier `init.ora` et `spfile.ora` (optionnel)

Une sauvegarde de tous les fichiers de la base lorsque la base de données est fermée permet d'obtenir une image complète de la base telle qu'elle existait au moment de son arrêt.





Une « copie » des fichiers avec la base de données ouverte ne serait pas valide, à moins de réaliser une sauvegarde en ligne !

Une sauvegarde hors ligne suivant un arrêt anormal de la base de données (ABORT) serait considérée comme incohérente.

⇒ Lors d'une restauration, son utilisation ne serait pas garantie et demanderait davantage d'efforts.

34.2 Sauvegarde base en ligne

Vous pouvez utiliser les sauvegardes en ligne pour n'importe quelle base de données qui fonctionne en mode ARCHIVELOG.

⇒ Dans ce mode, les fichiers de Redo Log en ligne sont archivés, ce qui génère un journal complet de toutes les transactions effectuées sur la base de données.

Les fichiers suivants doivent être sauvegardés :

- ◆ Fichier de contrôle
- ◆ Fichiers de données (sauvegardés à chaud)
- ◆ Fichier `init.ora` et `spfile.ora` (optionnel)



Une base de données peut être restaurée complètement à partir d'une sauvegarde en ligne, en ajoutant une récupération des données à partir des fichiers de Redo Log archivés.

Depuis la version 8i, Oracle propose un outil, le LOGMINER (package DBMS_LOGMNR), qui permet d'analyser les fichiers de Redo Log et de récupérer les ordres SQL de mise à jour exécutés par les transactions (ordres REDO) ainsi que les ordres SQL inverses (ordres UNDO).

Cet outil peut théoriquement être utilisé pour récupérer des ordres SQL permettant de rejouer les transactions pour une restauration incomplète.

Dans la pratique, la mise en œuvre n'est pas immédiate et le traitement de récupération peut être long.

34.2.1 Sauvegarde du fichier de contrôle

Le fichier de contrôle doit être sauvegardé lors de chaque sauvegarde complète ou partielle de la base de données.

Entre deux sauvegardes « normales », il est conseillé de sauvegarder le fichier de contrôle après toute restructuration importante de celle-ci (ajout/déplacement de fichiers de données ou de Redo Log)



Pour une sauvegarde base ouverte, utiliser l'ordre SQL : ALTER DATABASE :

```
ALTER DATABASE BACKUP CONTROLFILE TO  
[ nouveau nom | trace ] [ reuse ]  
;
```

En général, les sauvegardes binaires du fichier de contrôle sont suffisantes.

```
ALTER DATABASE BACKUP CONTROLFILE  
TO 'f:\oracle\backup\ORCL\control.bak' ;
```



Ne pas faire une copie du fichier de contrôle au niveau du système d'exploitation alors que la base est ouverte car celle-ci serait inexploitable.

34.3 Sauvegarde partielle d'un tablespace ONLINE

Lorsque le tablespace est passé en mode `BACKUP`, Oracle arrête d'écrire les *checkpoints* dans l'en-tête des fichiers de données du tablespace. Par contre, l'activité de lecture et d'écriture se poursuit normalement.

Une fois la sauvegarde terminée, le `END BACKUP` permet de sortir le tablespace du mode `BACKUP` et d'autoriser Oracle à reprendre les `CHECKPOINTS` dans l'en-tête des fichiers de données du tablespace.

Les fichiers sauvegardés sont incohérents mais Oracle a conservé la date et le numéro du dernier `CHECKPOINT` dans l'en-tête de chaque fichier. Ainsi, il pourra appliquer au fichier de données toutes les modifications postérieures au dernier `CHECKPOINT` (à partir des fichiers de Redo Log archivés) lors d'une restauration.



Cette sauvegarde n'est possible qu'en mode `ARCHIVELOG`. En supplément, il est important de faire une sauvegarde du fichier de contrôle et bien documenter ce qui a été sauvegardé.

Il est techniquement possible de mettre en parallèle des sauvegardes `ONLINE` de plusieurs tablespaces en mode `BACKUP`, mais ce n'est pas recommandé car l'activité sur les fichiers de Redo Log est augmentée.



Une sauvegarde `ONLINE` d'un tablespace qui n'est pas en mode `BACKUP` est généralement inexploitable.



L'oubli du `END BACKUP` ne génère aucun message de la part d'Oracle, pas même dans le fichier des alertes.

♦ **Ne pas faire**

```
Alter tablespace tbs1 begin backup ;
Alter tablespace tbs2 begin backup ;

•      sauvegarde des tablespaces -----

Alter tablespace tbs1 end backup ;
Alter tablespace tbs2 end backup ;
```

♦ **Faire plutôt**

```
Alter tablespace tbs1 begin backup ;
•      sauvegarde du tablespace tbs1 ---
Alter tablespace tbs1 end backup ;

Alter tablespace tbs2 begin backup ;
•      sauvegarde du tablespace tbs2 ---
Alter tablespace tbs2 end backup ;
```



Si un arrêt anormal de la base (ABORT) se produit alors qu'un tablespace est en mode BACKUP, une restauration du tablespace sera nécessaire au redémarrage

↗ à partir de la sauvegarde précédente du tablespace.

Un tablespace `READ ONLY` peut être sauvegardé `ONLINE` sans le mettre en mode `BACKUP` car il n'y a pas d'activité de mise à jour dans ce tablespace. Il n'y a pas de risque de données corrompues.

34.4 Sauvegarde de tous les tablespaces de la base ONLINE

La version 10g permet de placer tous les datafiles de la base de données dans le mode de sauvegarde `ONLINE` avec une seule commande. Vous n'avez plus besoin de placer chaque tablespace dans le mode de sauvegarde `ONLINE` un par un.

Pour cela, utiliser la commande :

```
ALTER DATABASE BEGIN BACKUP ;
-----
ALTER DATABASE END BACKUP ;
```

Pour placer tous les fichiers de la base de données dans le mode de sauvegarde `ONLINE`, la base doit être ouverte (`OPEN`) et en mode `ARCHIVELOG`.



Pendant la sauvegarde, il n'est plus possible d'exécuter un `SHUTDOWN` normal, de placer un tablespace dans mode `READ ONLY`, ou dans le mode sauvegarde en ligne (`ONLINE BACKUP MODE`) ou de mettre un tablespace `OFFLINE` avec les options habituelles.

Toutefois, quand vous exécutez la commande `ALTER DATABASE BEGIN BACKUP`, tous les fichiers inexistants, `OFFLINE` ou `READ ONLY`, sont ignorés et le processus continue.

Si vous avez un « datafile » avec un statut `OFFLINE` un message d'avertissement est affiché.

34.5 Vues du dictionnaire de données

Plusieurs vues du dictionnaire permettent d'obtenir des informations sur l'archivage :

- `V$DATABASE` : mode de fonctionnement de la base (colonne `LOG_MODE`)
- `V$INSTANCE` : statut du processus `ARCH` (colonne `ARCHIVER`)
- `V$LOG` : statut des groupes vis à vis de l'archivage (colonne `ARCHIVED`)
- `V$ARCHIVED_LOG` : informations sur les fichiers de Redo Log archivés
- `V$ARCHIVED_DEST` : informations sur les destinations d'archivage

V\$ARCHIVED_LOG	
RECID	Identifiant de l'enregistrement
NAME	Chemin complet de l'archive
SEQUENCE#	Numéro de séquence du fichier de Redo Log correspondant
FIRST_CHANGE#	Plus petit numéro de SCN (numéro de transaction) écrit dans l'archive
FIRST_TIME	Date et heure du plus petit numéro de SCN
NEXT_CHANGE#	Plus grand numéro SCN écrit dans l'archive
NEXT_TIME	Date et heure du plus grand numéro de SCN
COMPLETION_TIME	Date et heure de l'archivage

V\$ARCHIVED_DEST	
DEST_NAME	Nom de la destination
DESTINATION	Chemin complet de la destination
STATUS	Statut de la destination (<code>VALID</code> , <code>ERROR</code> , ...)
ERROR	Message d'erreur (en cas d'erreur)



34.6 Stratégie recommandée par Oracle

Le Grid Control permet de définir une stratégie de sauvegarde recommandée par Oracle, qui permet de protéger les données et fournit des possibilités de recouvrement de la base de données pour les 24 heures.

La stratégie recommandée par Oracle utilise la sauvegarde incrémentale et les caractéristiques de mises à jour des sauvegardes incrémentales proposées dans Recovery Manager (*RMAN*), en fournissant un processus de recouvrement plus rapide que si l'on applique les archives de Redo log.

La stratégie Oracle prend une copie complète de la base de données pour la première sauvegarde.

Il s'agit d'une sauvegarde complète de la base, suivie de sauvegardes incrémentales sur disque effectuées tous les jours à des heures précises via *RMAN*.

Comme ces sauvegardes sur disque sont mémorisées, vous pouvez toujours effectuer un recouvrement complet de la base ou un recouvrement de l'image faite à un moment donné le jour d'avant.

Une fois que vous avez complété et accepté les options de l'écran, votre base sera automatiquement sauvegardée.

34.7 Recover Manager (RMAN)

RMAN (*Recovery Manager*) est un outil qui accomplit une bonne partie du travail à la place du DBA dans le but de protéger la base de données.

Des opérations qui demandent du temps et qui sont délicates à réaliser sont exécutées avec *RMAN* au moyen de quelques commandes.

Lors de l'emploi de l'utilitaire *RMAN* (*Recovery Manager*), vous n'avez pas besoin de placer explicitement chaque tablespace dans un état de sauvegarde.

RMAN lit les blocs de données de la même manière qu'Oracle le fait lors de requêtes. Il n'y a plus aucun risque de corruption de fichier sauvegardé.

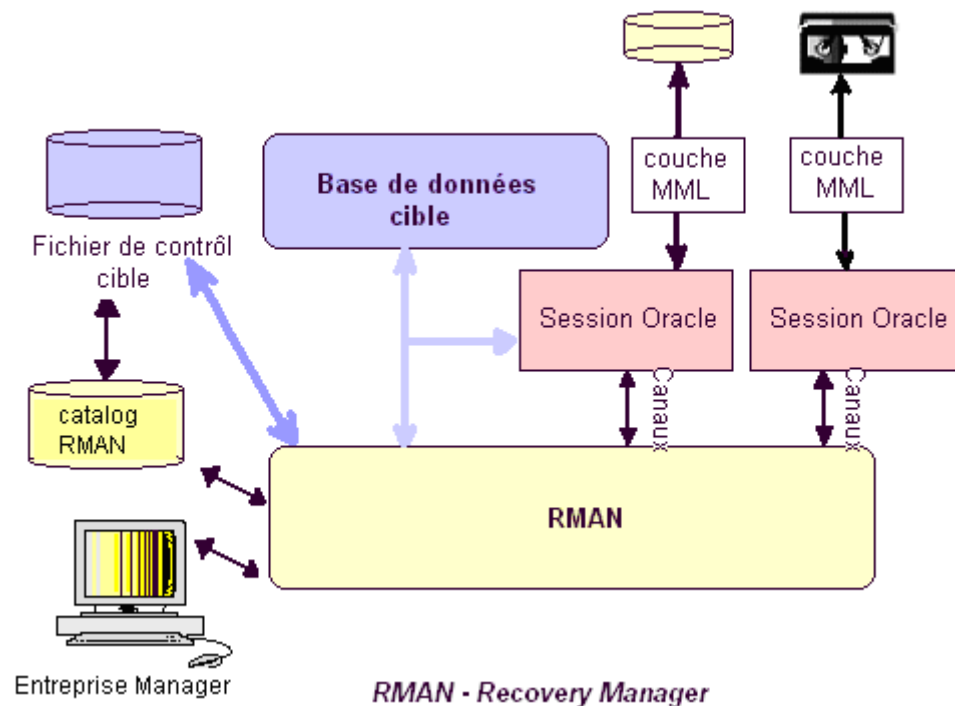
RMAN peut effectuer de nombreuses opérations identiques pour la plupart à celles que vous pouvez réaliser manuellement :

- ♦ Copie de fichiers de bases de données :
RMAN peut créer des copies images de fichiers de données ou de contrôle ce qui revient à accomplir une sauvegarde à chaud.
- ♦ Création d'une base de données dupliquée :
RMAN peut créer une copie de votre base de données sur la même machine ou sur une machine différente. Cette base peut avoir le même nom que votre base actuelle ou un nom différent.
- ♦ Création d'une base de données de secours :
RMAN peut créer une base de données de secours qui servira à reprendre la fonction de base principale en cas de défaillance de cette dernière.
Récupération de tablespace jusqu'à un point donné dans le temps : *TSPITR* (Tablespace Point-In-Time Recovery)
RMAN peut effectuer une récupération d'un tablespace jusqu'à un point dans le temps. Il est possible ainsi de limiter la perte des données en rétablissant celui-ci jusqu'au point précédant l'incident. Les autres parties de votre base restent actuelles.



Exemple de sauvegarde RMAN

```
connect target
run {
configure device type DISK parallelism 1;
configure controlfile autobackup on;
configure controlfile autobackup format for device type disk to
'C:\sav_ora\autobackup\MABASE\SPFCTL_%d_%T_%F.BCK';
sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
sql 'ALTER SYSTEM CHECKPOINT';
backup as compressed backupset archivelog
from time 'sysdate-7' until time 'sysdate'
format 'C:\sav_ora\arch\MABASE\Arch_%d_%T_%s%p'
tag = ARCH_MABASE
maxsetsize 4G ;
backup as compressed backupset database
format 'C:\sav_ora\db\MABASE\DBfull_%d_%T_%s%p'
tag = MABASE'
MAXSETSIZE 4G
FILESERSET=3
include current controlfile;
sql 'ALTER SYSTEM CHECKPOINT';
sql 'ALTER SYSTEM ARCHIVE LOG CURRENT';
sql 'ALTER DATABASE BACKUP CONTROLFILE TO TRACE';
crosscheck backup device type disk;
delete expired backup;
delete obsolete redundancy 7 device type disk;
backup current controlfile format 'C:\sav_ora\db\MABASE\CTL_%d_%T_%s%p';
}
```



34.8 Le Flash Back

Le flashback permet de récupérer un ensemble de données ou d'objets dans le passé puis de les réinjecter dans la base de données. La technologie d'Oracle 10g ou 11g, offre la capacité d'interroger des versions anciennes de schéma d'objets ou de données.

Le « *flashback* » a été créé pour réparer facilement les données corrompues d'une table par un batch, en réinjectant dans la base de données les données récupérées avant le passage du batch grâce au flashback. Le *flashback* permet un retour arrière dans la base de données afin de sélectionner des objets ou parties d'objets pour les réinjecter dans la version actuelle de la base de données.

La version 9i introduisait la notion de « *flashback query* » pour fournir un mécanisme simple afin de réparer les erreurs humaines.

Oracle 10g et 11g étendent la technologie *flashback* pour assurer vite et facilement une réparation à tous les niveaux :

- ♦ **Flashback database** ; vous laisse rapidement ramener votre base à un point dans le temps en réparant toutes les modifications apportées depuis cet instant.
- ♦ **Flashback table** ; vous permet de retrouver rapidement une table et son contenu à un moment dans le passé.
- ♦ **Flashback Query** ; vous laisse voir les modifications apportées par une transaction à une ou plusieurs données, accompagnées de ses métadonnées.

Objet	Scénario	Flashback
Database	DROP USER	Flashback database
	TRUNCATE TABLE	Flashback database
	Jobs Batches	Flashback database
Table	DROP TABLE	Flashback Table
	UPDATE avec clause WHERE erronée	Flashback Table
Transaction	Comparer des données passées avec des données actuelles	Flashback Transaction
	Exécuter un job 2 fois car on n'est pas sûr de la validité des objets	Flashback Transaction



35 Procédures de sauvegardes

Il existe plusieurs types de sauvegarde d'une base de données Oracle. Il est donc possible de se prémunir contre un incident suite à une panne.

- ♦ Sauvegardes logiques (EXPORT)
- ♦ Sauvegardes Physiques Hors Ligne (à froid)
- ♦ Sauvegardes Physiques en Ligne (à chaud)

Les sauvegardes hors ligne et les sauvegardes en ligne sont des sauvegardes physiques de fichiers. Celles-ci peuvent être réalisées au moyen de scripts ou de l'utilitaire `RMAN`.

Une stratégie de sauvegarde solide doit comprendre des sauvegardes physiques et des sauvegardes logiques.

Selon les caractéristiques de votre base, une des méthodes devrait être choisie en tant que procédure principale, et au minimum une autre méthode devrait être utilisée en tant que méthode de secours.

Une sauvegarde physique consiste en la copie des fichiers qui constituent la base de données.

Oracle supporte deux types de sauvegardes physiques :

- ⇒ Sauvegarde hors ligne appelée encore sauvegarde à froid (base arrêtée)
- ⇒ Sauvegarde en ligne appelée encore sauvegarde à chaud (base ouverte)

35.1 Sauvegardes logiques

Une sauvegarde logique de la base de données consiste en la lecture des données de la base de données à l'aide d'utilitaires oracle, dans le but d'écrire ces données dans un fichier binaire, cette lecture se déroule indépendamment de l'emplacement physique des données.

IMPORT/EXPORT

Oracle fournit les utilitaires `IMPORT/EXPORT` pour réaliser ce genre de sauvegardes.

L'utilitaire `EXPORT` permet de lire les données de la base de données pour en extraire les données et les mettre dans un fichier binaire.

L'outil `IMPORT` réalise l'opération inverse de récupération des données du fichier binaire généré. Celui-ci lit le fichier créé par l'`EXPORT` et exécute les commandes qu'il y trouve.

- ⇒ Par exemple ; `CREATE TABLE` puis `INSERT` pour charger les données dans la table

Il est possible d'Exporter :

- ⇒ une base de données complète
- ⇒ le contenu d'un tablespace
- ⇒ des utilisateurs ou schémas
- ⇒ certaines tables seulement



Lors des opérations vous pouvez choisir d'Exporter ou non les informations du dictionnaire de données associées aux tables, telles que les privilèges ou les index et contraintes d'intégrités.

Les données Exportées d'une base Oracle peuvent être importées dans une base créée avec une version immédiatement supérieure du système. La possibilité inverse est également supportée mais des actions supplémentaires seront nécessaires pour employer l'ancienne version des vues utilisées par l'Export.

- ♦ Le fichier `readme.doc` qui accompagne la version du produit concerné décrit les exigences à respecter pour l'emploi de ces utilitaires avec différentes versions spécifiques

L'utilitaire d'EXPORT d'Oracle lit la base de données, y compris le dictionnaire de données écrit des informations dans un fichier binaire de transfert appelé « fichier d'Export », ou « fichier DUMP ».

Data pump est un nouvel outil qui permet de charger ou décharger des données à grande vitesse.

Il peut être appelé via le package PL/SQL, `DBMS_DATAPUMP`. Il reprend les fonctionnalités des outils IMPORT/EXPORT en plus puissant.

Ces nouveaux outils proposent :

- ♦ De nouvelles commandes pour Export et Import appelés respectivement `EXPDP` et `IMPDP`.
- ♦ Une interface Web d'Import et d'Export accessible à partir du Database Control est disponible.
 - `EXPORT DATA Pump` (via des tables externes)
 - `IMPORT DATA Pump` (via des tables Externes)
 - `SQL*Load`

Flashback

Le flashback permet la récupération d'un ensemble de données ou d'objets dans le temps puis de les réinjecter dans la base de données.

La technologie de la mémoire d'oracle 10g, offre la capacité d'interroger des versions anciennes de schéma des objets et d'interroger les données historiques.

La version 9i introduisait la notion de « *flashback query* » pour fournir un mécanisme simple pour réparer les erreurs humaines.

Oracle 10g étend la technologie *flashback* pour assurer vite et facilement une réparation à tous les niveaux :

- ♦ **Flashback database** : vous laisse rapidement ramener votre base à un point dans le temps en réparant toutes les modifications apportées depuis cet instant.
- ♦ **Flashback drop** : donne une solution pour restaurer accidentellement des tables
- ♦ **Flashback table** : vous permet de retrouver rapidement une table et son contenu à un moment dans le passé.
- ♦ **Flashback Query** : vous laisse voir les modifications apportées à une ou plusieurs données accompagnées de ses métadonnées.



35.2 Sauvegardes hors ligne

Les sauvegardes hors lignes se produisent lorsque la base a été arrêtée proprement.

⇒ **Pas de panne instance ou shutdown abort !**

Les fichiers suivants doivent être sauvegardés :

- ◆ Fichiers de contrôle
- ◆ Fichiers de données
- ◆ Fichiers de redo log en ligne
- ◆ Fichiers `init.ora` et `spfile.ora`

Une sauvegarde de tous les fichiers de la base lorsque la base de données est fermée permet d'obtenir une image complète de la base de données telle qu'elle existait au moment de la sauvegarde.

Une sauvegarde hors ligne suivant un arrêt anormal de la base de données (`ABORT`) serait considérée comme incohérente, lors d'une restauration, son utilisation ne serait pas garantie et demanderait davantage d'efforts



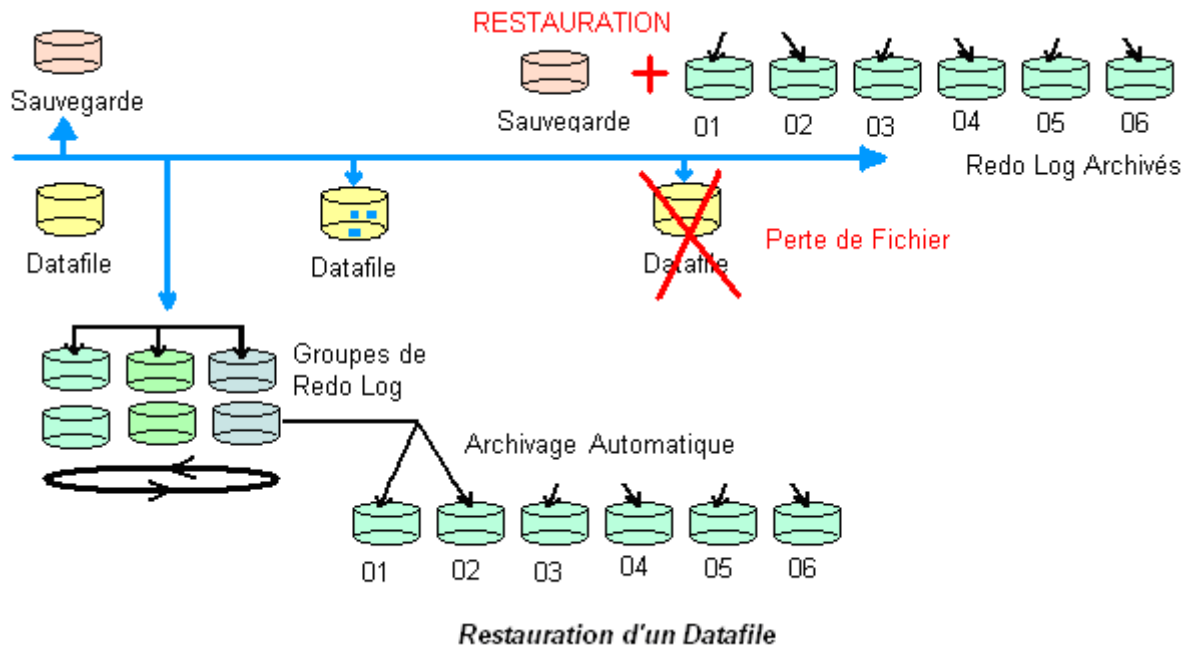
Une sauvegarde du système de fichiers avec la base de données ouverte ne serait pas valide, à moins de réaliser une sauvegarde en ligne (Oracle ne supporte pas que l'on bouge un fichier pendant qu'il écrit dedans !)

35.3 Sauvegardes en ligne

Vous pouvez utiliser les sauvegardes en ligne pour n'importe quelle base de données qui fonctionne en mode `ARCHIVELOG`.

Dans ce mode les fichiers de redo log sont archivés, ce qui génère un journal complet de toutes les transactions effectuées sur la base de données.





Lorsque la base de données est ouverte, les fichiers suivants peuvent être sauvegardés :

- ◆ Les fichiers de données
- ◆ Les fichiers de redo log archivés
- ◆ Un fichier de contrôle au moyen de la commande `alter database`
- ◆ Un export du fichier de paramètre `SPFILE`.

Les procédures de sauvegarde en ligne sont très puissantes :

- ⇒ Elles offrent des possibilités de récupérations complètes jusqu'à un point donné dans le temps
- ⇒ Elles permettent à la base de rester ouverte durant la sauvegarde du système de fichiers

Ainsi les bases qui ne peuvent pas être arrêtées en raison de besoins utilisateurs peuvent faire l'objet de sauvegardes de leur système de fichiers.

- ⇒ Cela réduit le nombre d'opérations d'entrées/sorties physiques requises par la base

Maintenir la base de données ouverte évite que la zone SGA (*Système Global Area*) de l'instance soit réinitialisée comme c'est le cas lors de chaque démarrage de la base.

Empêcher la réinitialisation de la mémoire améliore les performances ;

La plupart des bases de données de production, en particulier celles qui supportent les applications transactionnelles, doivent opérer en mode `ARCHIVELOG`.



Vous pouvez effectuer une sauvegarde du système de fichiers d'une base de données ouverte si elle fonctionne en mode `ARCHIVELOG`.

- ◆ Placer chaque tablespace dans un « état de sauvegarde »
- ◆ Sauvegarder les fichiers de données
- ◆ Rétablir les tablespaces dans un état normal



Une base de données peut être restaurée complètement à partir d'une sauvegarde en ligne, en ajoutant une récupération des données à partir des fichiers de redo log archivés.

Depuis la version 8i Oracle propose un outil le `LOGMINER` (package `DBMS_LOGMNR`), qui permet d'analyser les fichiers de Redo Log et de récupérer les ordres SQL de mise à jour exécutés par les transactions (ordres `REDO`) ainsi que les ordres SQL inverses (ordres `UNDO`).

Cet outil peut théoriquement être utilisé pour récupérer des ordres SQL permettant de rejouer les transactions pour une restauration incomplète.

Dans la pratique, la mise en œuvre n'est pas immédiate et le traitement de récupération peut être long.

35.4 RMAN

`RMAN` est un outil qui accomplit une bonne partie du travail à la place du DBA dans le but de protéger la base de données.

Des opérations qui demandent du temps et qui sont délicates à réaliser sont exécutées avec `RMAN` au moyen de quelques commandes. Par exemple, lors de l'emploi de l'utilitaire `RMAN` (*Recovery Manager*), vous n'avez pas besoin de placer explicitement chaque tablespace dans un état de sauvegarde, `RMAN` lit les blocs de données de la même manière qu'Oracle le fait lors de requêtes.

`RMAN` peut effectuer de nombreuses opérations identiques pour la plupart à celles que vous pouvez réaliser manuellement :

- ⇒ Sauvegardes ou Copie de fichiers de bases de données
`RMAN` peut créer des sauvegardes ou des copies images de fichiers de données ou de contrôle, ce qui revient à accomplir une sauvegarde à chaud
- ⇒ Création d'une base de données dupliquée
`RMAN` peut créer une copie de votre base de données sur la même machine ou sur une machine différente. Cette base peut avoir le même nom que votre base actuelle ou un nom différent.
- ⇒ Création d'une base de données de secours
`RMAN` peut créer une base de données de secours qui servira à reprendre la fonction de base principale en cas de défaillance de cette dernière.
- ⇒ Récupération de tablespace jusqu'à un point donné dans le temps : `TSPITR` (*Tablespace Point-In-Time Recovery*)



RMAN peut effectuer une récupération d'un tablespace jusqu'à un point dans le temps. Il est possible ainsi de limiter la perte des données en rétablissant celui-ci jusqu'au point précédant l'incident. Les autres parties de votre base restent actuelles.

35.5 Planification et intégration de procédures

Lors de la planification des sauvegardes physiques, envisagez également la possibilité d'utiliser l'utilitaire RMAN pour effectuer des sauvegardes incrémentielles.

Dans ce chapitre nous allons voir :

- ⇒ Comment intégrer des sauvegardes logiques et physiques
- ⇒ Comment intégrer des sauvegardes de la base de données avec des sauvegardes du système d'exploitation.

La procédure de sélection de la méthode de sauvegarde principale doit prendre en compte les caractéristiques de chaque méthode.

Méthode	Type	Caractéristiques de récupération
Export & DATA Pump &Flashback	Logique	Permet de récupérer n'importe quel objet de la base de données dans un état où il se trouvait lors de l'export ou dans le temps.
Sauvegarde hors ligne	Physique	Permet de récupérer la base de données dans l'état où elle se trouvait au moment où elle a été arrêtée. Si elle opère dans le mode ARCHIVELOG, vous pouvez rétablir son état jusqu'à n'importe quel point dans le temps.
Sauvegarde en ligne	Physique	Permet de récupérer la base jusqu'à n'importe quel point dans le temps.

Les sauvegardes hors ligne sont les moins souples lorsque la base opère dans le mode NOARCHIVELOG. Elles représentent un « *cliché* » à un moment donné.

Etant donné que ce sont des sauvegardes physiques, les DBA ne peuvent pas procéder à une récupération sélective d'objets logiques (par exemple une table).

Elles doivent être utilisées normalement comme moyen vers lequel se tourner lorsque la méthode de sauvegarde principale échoue.

Si la base fonctionne en mode ARCHIVELOG, vous pouvez utiliser les sauvegardes hors ligne comme base pour une récupération de média.



Dans une telle situation, pourquoi ne pas procéder à une sauvegarde en ligne.

Sur les deux méthodes qui restent laquelle est la plus appropriée :

- ⇒ Sauvegarde en ligne
- ⇒ Sauvegarde logique

Pour les environnements en production la réponse est presque toujours : la sauvegarde en ligne.

Celle-ci permet de récupérer la base de données jusqu'au moment ayant précédé l'erreur système ou l'erreur utilisateur.

L'emploi d'une stratégie fondée sur `EXPORT` (ou `DATA Pump`) permet de récupérer les objets uniquement dans l'état où ils étaient au moment de l'`EXPORT`.

Vous devez prendre en compte :

- ◆ la taille de la base
- ◆ l'importance des objets à récupérer.

Vous devez également vous demander combien de temps prendra une récupération des données dans le cas d'un scénario d'incident standard tel que la perte d'un disque.

Lorsqu'un fichier est perdu, la méthode la plus rapide de rétablissement passe en général par l'emploi d'une sauvegarde physique ce qui favorise une fois de plus les sauvegardes en ligne par rapport aux `EXPORTS`.



On ne peut pas appliquer des archives de Redo Logs après un IMPORT !

Si la base est petite, que le volume de transactions est très faible et que la disponibilité de la base ne soit pas un problème, les sauvegardes hors ligne et les `EXPORTS` pourront répondre à vos besoins.

Si vous êtes uniquement préoccupé par 1 ou 2 tables, utilisez des `EXPORTS` sélectifs.

Si la base est grande, le temps de récupération requis par l'emploi des utilitaires `IMPORT/EXPORT`, peut être prohibitif.

Pour de grands environnements à faibles volumes de transactions, les sauvegardes hors lignes peuvent être appropriées.

Indépendamment du choix de la méthode principale de sauvegarde, l'implémentation finale doit inclure une sauvegarde physique et une sauvegarde logique, soit au moyen d'`EXPORT` ou au moyen de la réplication.

Cette redondance est nécessaire car ces méthodes garantissent différents aspects de la base :

- ⇒ `EXPORT` Garantit la cohérence logique des données
- ⇒ Sauvegardes physiques garantissent la cohérence physique de la base



Exemple

Type de Base de Données	Sauvegarde en ligne	Sauvegarde hors ligne	EXPORT
Toutes tailles avec transactions intensives	Nocturne	Hebdomadaire	Hebdomadaire
Petite, principalement en lecture seule	Aucune	Nocturne	Nocturne
Grandes, principalement en lecture seule	Aucune	Nocturne	Hebdomadaire

La fréquence et le type de sauvegardes varient selon les caractéristiques d'exploitation de la base de données.

D'autres activités de la base peuvent requérir des procédures de sauvegarde :

- ♦ Sauvegardes hors lignes réalisées avant des mises à jour de la base
- ♦ Sauvegardes avec EXPORT lors de migration d'une application entre deux bases de données

35.6 Intégration des sauvegardes Oracle et du système

Les activités de sauvegarde liées à la fonction de DBA impliquent un certain nombre de tâches qui incombent normalement au groupe d'administrateurs système :

- ♦ Suivi de l'utilisation des disques
- ♦ Maintenance des bandes
- ♦ Etc ...

Au lieu de dupliquer ces actions, il est préférable de les coordonner en adoptant une approche s'alignant sur les processus, afin d'aboutir à une meilleure intégration de l'ensemble des tâches.

La stratégie de sauvegarde de la base de données doit être modifiée de façon à ce que les administrateurs système qui réalisent les sauvegardes système se chargent de toute la maintenance des bandes,

- ⇒ afin de vous permettre de centraliser les processus de contrôle de la protection de votre environnement de production

La centralisation des processus de contrôle est en général réalisée en dédiant certains disques à l'hébergement des sauvegardes physiques des fichiers de la base de données.

Au lieu de les sauvegarder sur bandes, elles sont écrites sur d'autres disques du même serveur.

Le contenu de ces disques doit être pris en compte par les sauvegardes régulières systèmes effectuées par les administrateurs system.

Le DBA n'a pas besoin d'exécuter une tâche de sauvegarde sur bande séparée mais il doit toutefois veiller à ce que ces dernières se soient bien déroulées et aient pu se terminer correctement.



35.7 Les grandes bases de données

Souvent vous pouvez recharger la base de données et recréer les agrégats de données en moins de temps qu'il en faudrait pour restaurer les données en employant les utilitaires de sauvegarde et de récupération d'Oracle.

Si vous pouvez recharger les données plus rapidement que vous ne les restaurez, on peut se demander s'il est bien utile de les sauvegarder ?

La réponse dépend de la façon dont le traitement de chargement intervient.

Ces bases de données possèdent 4 grands types de tables :

- ♦ Grandes Tables de transactions qui contiennent la majorité des données brutes de la base
- ♦ Tables d'agrégats, qui conservent des données agrégées des grandes tables de transactions
- ♦ Tables de codes
- ♦ Tables de travail temporaire

Lors de l'évaluation des besoins de sauvegarde de votre base, vous devrez évaluer les besoins de chaque type de tables.

Grandes tables de transactions

Les exigences de sauvegardes des grandes tables de transactions sont conditionnées par les méthodes de chargement des données utilisées, à savoir :

- ⇒ Si lors de chaque chargement, les données sont complètement remplacées, vous pouvez utiliser le processus de chargement pour restaurer les données
- ⇒ Si ces données ne sont pas complètement remplacées, vous devez utiliser une combinaison des méthodes de sauvegarde pour pouvoir récupérer les données en cas de besoin.

Exemple

Chaque chargement de données dans une table d'informations de ventes concerne une période précise, vous aurez besoin de sauvegarder les données de la période précédente, ainsi que celles de la période actuelle

Vous disposez de 2 méthodes :

- ♦ Sauvegarder les données anciennes et utiliser le processus de chargement pour recréer les données de la période actuelle.
- ♦ Sauvegarder les anciens fichiers de données puis, lors d'une procédure de récupération, exécuter le chargement de chaque période séparément.

Selon les procédures de chargement de données que vous utilisez, la seconde peut permettre une récupération plus rapide des données.



Si les données de vos tables de transactions peuvent faire l'objet de mises à jour après leur chargement, vous devriez être en mesure de pouvoir recréer ces transactions.

Vous pouvez soit :

- ♦ Exporter les données consécutivement aux transactions de modifications
- ♦ Exécuter les transactions avec une base en mode ARCHIVELOG (la table ne peut plus être placée en mode NOLOGGING, mais vous pouvez utiliser RMAN pour réduire la quantité de données sauvegardées en utilisant des sauvegardes incrémentales)



LOGGING spécifie que la création de l'index sera écrite dans les Redo Logs. Si la base est en NOARCHIVELOG, la création d'index n'est pas écrite en Redo Log, même si LOGGING est spécifié.

Vous pouvez sauvegarder les données après chaque chargement, ce qui est une méthode de sauvegarde supplémentaire.

Vous devrez sauvegarder les transactions utilisateurs ou les anciennes données :

- ♦ Si les données peuvent être modifiées après chargement
- ♦ Si un chargement ne concerne pas toutes les données

Tables d'agrégats

Elles stockent des données redondantes.

Toutes les données de ces tables peuvent être régénérées en exécutant de nouveau les transactions utilisées pour les créer.

- ⇒ Vous ne devriez pas autoriser de modifications directes sur ces tables après leur création.
- ⇒ En cas de besoin, modifiez les tables de transactions représentant la source des tables d'agrégat puis recréez ces dernières

Ainsi il n'est pas utile de sauvegarder ces tables d'agrégats

Vous pouvez exporter les données après leur création, mais il est en général plus rapide de recréer ces dernières en cas de problème.

Tables de code

Elles conservent des données de type statistique.

Elles ne devraient faire l'objet que de très peu de transactions.

Une protection au moyen d'EXPORT est, en général, suffisante.

Comme très peu de transactions les concernent, il n'est habituellement pas nécessaire d'utiliser les fichiers de redo log archivés pour les restaurer.

Si vous planifiez correctement les opérations d'EXPORT, vous devriez être en mesure de les utiliser lors d'une récupération sans perte de données.



Tables temporaires

Elles sont utilisées lors du traitement de chargement de données ; en général il n'est pas prévu de pouvoir y accéder à partir de l'application en production.

Les seuls besoins de récupération pour ces tables, pourraient provenir directement de la procédure de chargement des données.

Comme aucune transaction en ligne ne porte sur ces tables, les `EXPORTS` sont généralement utilisés pour les sauvegarder.

Exemple

Vous pourriez souhaiter sauvegarder le contenu de ces tables à différentes étapes de la procédure de chargement, afin de réduire au maximum l'activité de récupération nécessaire en cas d'échec de la procédure.



36 Restaurations

Dans une restauration, **c'est l'enchaînement des actions qui est compliqué**, pas la commande RECOVER en elle-même.

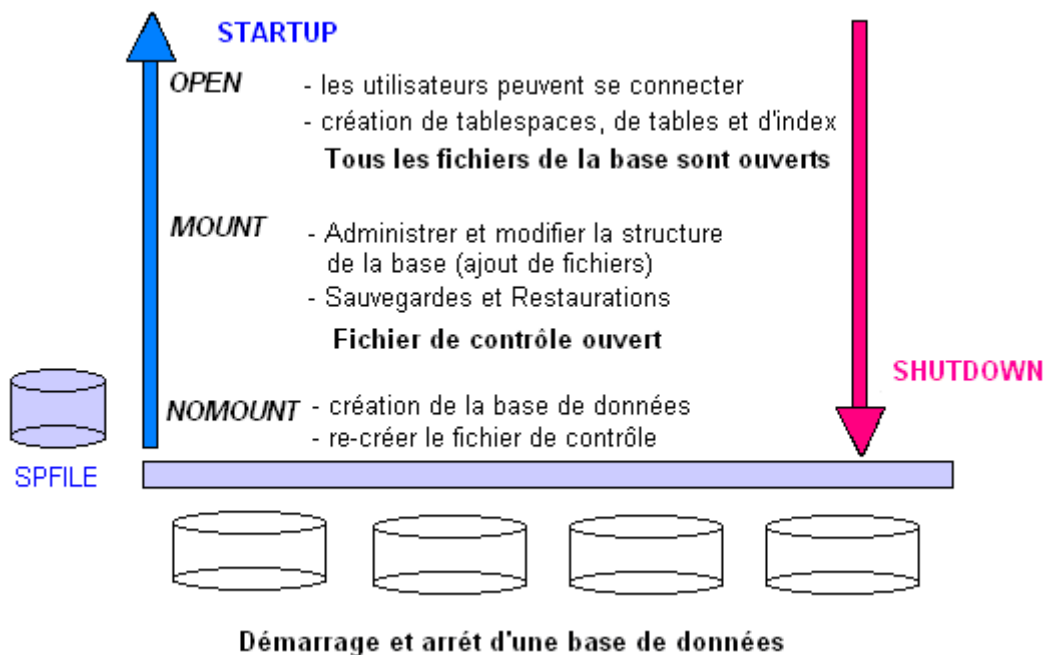


NE PAS SE PRECIPITER.

Avant de commencer toute opération de restauration, faire si possible une sauvegarde complète de la base endommagée. Elle fournit un point de retour en cas d'aggravation de la situation par une mauvaise manipulation.

⇒ Toutes les opérations de restauration nécessitent le privilège SYSDBA.

Lors d'un problème de perte de fichier, la base de données reste bloquée à l'état NOMOUNT s'il s'agit du fichier de contrôle, ou à l'état MOUNT s'il s'agit d'un fichier de données.



36.1 La commande RECOVER

Il existe trois variantes de la commande RECOVER :

- ⇒ **RECOVER DATABASE** : Tous les fichiers de la base qui nécessitent une restauration sont traités.
- ⇒ **RECOVER TABLESPACE** : Tous les fichiers des tablespaces cités en paramètre qui nécessitent une restauration, sont traités.
- ⇒ **RECOVER DATAFILE** : Seuls les fichiers cités en paramètre, s'ils nécessitent une restauration, sont traités.



```
RECOVER [AUTOMATIC] [FROM 'location']  
DATABASE [ UNTIL CANCEL  
| UNTIL TIME 'YYYY-MM-DD:HH24:MI:SS'  
| UNTIL CHANGE integer ]  
[ USING BACKUP CONTROLFILE ]
```

Dans les trois cas, la commande `RECOVER` est capable de déterminer seule, quels sont les fichiers de Redo Log à appliquer sur les différents fichiers lors de la restauration.

Elle recherche les fichiers de Redo Log archivés qu'elle souhaite appliquer dans le répertoire défini par la valeur actuelle du paramètre `LOG_ARCHIVE_DEST_1` du fichier `spfile`.

Si les fichiers de Redo Log archivés ne sont pas à l'emplacement attendu (soit parce que la destination des archives a changé, soit parce que les archives ont été archivées sur bande), il faudra intervenir pour aider la commande `RECOVER` à trouver l'emplacement.

Par défaut, la commande `RECOVER` fonctionne en mode manuel ; elle demande confirmation de l'emplacement de chaque fichier de Redo Log archivé qu'elle s'apprête à appliquer.

Il est possible alors :

- ♦ de valider (pour indiquer que l'archive est bien à l'emplacement indiqué)
- ♦ d'indiquer un autre emplacement si l'archive n'est pas/plus à l'endroit attendu
- ♦ de taper `AUTO` pour passer en mode automatique (voir ci-dessous) ou de taper `CANCEL` pour arrêter l'opération

36.1.1 Exemples de restaurations

- ♦ **AUTOMATIC** = la restauration se fera de façon automatique sans demander à l'opérateur de confirmer le passage de chaque fichier d'archive.

```
SQL> RECOVER DATABASE  
ORA-00279: change 44629 generated at 08/07/2001 16:15:55 needed for thread 1  
ORA-00289: suggestion : D:\ORACLE\ORADATA\ORCL\ARCH\ORCL00002.ARC  
ORA-00280: change 44629 for thread 1 is in sequence #2  
Specify log: {<RET>=>suggested | filename | AUTO | CANCEL}  
  
--- réponse saisie ---  
d:\oracle\oradata\ORCL\temp\ORCL00002.arc
```

- ♦ **FROM 'location'** = précise à Oracle l'emplacement des archives de Redo Log

```
SQL> RECOVER DATABASE FROM 'd:\oracle\oradata\test\temp'  
ORA-00279: change 44629 generated at 08/07/2001 16:15:55 needed for thread 1  
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00002.ARC  
ORA-00280: change 44629 for thread 1 is in sequence #2  
  
ORA-00279: change 45013 generated at 08/08/2001 06:59:49 needed for thread 1  
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00003.ARC  
ORA-00280: change 45013 for thread 1 is in sequence #3  
ORA-00278: log file 'D:\ORACLE\ORADATA\TEST\TEMP\TEST00002.ARC' no longer needed for this  
recovery  
  
Log applied.  
Media recovery complete.  
SQL>
```

- ♦ **UNTIL CANCEL** = Précise à Oracle que la restauration s'arrête lorsque l'opérateur saisit `CANCEL`



```
SQL> RECOVER DATABASE
ORA-00279: change 44629 generated at 08/07/2001 16:15:55 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\ORCL\ARCH\ORCL00002.ARC
ORA-00280: change 44629 for thread 1 is in sequence #2
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

--- réponse saisie ---
d:\oracle\oradata\ORCL\temp\ORCL00002.arc

ORA-00279: change 45013 generated at 08/08/2001 06:59:49 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\ORCL\ARCH\ORCL00003.ARC
ORA-00280: change 45013 for thread 1 is in sequence #3
ORA-00278: log file 'd:\oracle\oradata\ORCL\temp\ORCL00002.arc' no longer needed for this
recovery
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

--- réponse saisie ---
cancel

Media recovery cancelled.
```

- ♦ **UNTIL TIME** = Précise la date et l'heure d'arrêt de la restauration, en cas de restauration partielle. La condition d'arrêt est spécifiée par une date et une heure sous forme d'une chaîne entre apostrophes au format YYYY-MM-DD:HH24:MI:SS. La fonction TO_DATE peut être utilisée pour la mise en forme de la date :
set until time "TO_DATE('02/24/2004 13:13:00','MM/DD/YYYY HH24:MI:SS')"
SQL> recover until time '2004-01-26:11:08:18';
ORA-00279: changement 72152 gŭnŭrŭ Ó 01/25/2004 15:51:23 requis pour thread 1
ORA-00289: suggestion : D:\ORACLE9\ADMIN\COLLEGE\ARCH\REDO000023.ARC
ORA-00280: le changement 72152 pour le thread 1 se trouve au no de sŭquence 23

- ♦ **UNTIL CHANGE integer**= Précise le numéro de transaction (SCN) d'arrêt en cas de restauration partielle. La condition d'arrêt est spécifiée par un numéro de changement (numéro de transaction).
- ♦ **USING BACKUP CONTROLFILE** = indique à Oracle qu'une sauvegarde du fichier de contrôle est utilisée.

L'opération RECOVER peut être arrêtée temporairement puis reprise ultérieurement ; dans ce cas, elle redémarre à l'endroit où elle s'était arrêtée.



Si le RECOVER est interrompu avant la fin, la base ne peut pas être ouverte. Il est obligatoire, soit d'aller jusqu'au bout, soit de demander explicitement une restauration incomplète avec l'option UNTIL.

Cette séquence montre la possibilité de désigner un autre emplacement pour les archives, la possibilité d'arrêter l'opération en cours de route et l'impossibilité dans ce cas d'ouvrir la base (la restauration n'est pas complète).



```
SQL> RECOVER DATABASE
ORA-00279: change 44629 generated at 08/07/2001 16:15:55 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\ORCL\ARCH\ORCL00002.ARC
ORA-00280: change 44629 for thread 1 is in sequence #2
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

--- réponse saisie ---
d:\oracle\oradata\ORCL\temp\ORCL00002.arc

ORA-00279: change 45013 generated at 08/08/2001 06:59:49 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\ORCL\ARCH\ORCL00003.ARC
ORA-00280: change 45013 for thread 1 is in sequence #3
ORA-00278: log file 'd:\oracle\oradata\ORCL\temp\ORCL00002.arc' no longer needed for this
recovery
Specify log: {<RET>=suggested | filename | AUTO | CANCEL}

--- réponse saisie ---
cancel
Media recovery cancelled.
SQL> ALTER DATABASE OPEN ;
```



Les fichiers de Redo Log archivés attendus par Oracle peuvent être identifiés grâce à la vue V\$RECOVERY_LOG !

Un RECOVER lancé en mode manuel, peut aussi être passé en mode automatique en tapant AUTO sur l'invite.

Si toutes les archives sont à un autre endroit que celui attendu, ou ramenées d'une bande à un autre endroit que celui attendu, il est possible de spécifier cet emplacement puis de lancer le RECOVER en mode automatique.

Le RECOVER utilise alors l'emplacement indiqué comme source des archives, ne demande pas confirmation et applique les archives.

Une source différente de l'emplacement attendu pour les fichiers de Redo Log archivés peut être indiquée (par exemple l'emplacement des fichiers d'archives) :

- ◆ Dans la commande RECOVER :

```
RECOVER FROM 'd:\temp' DATABASE
;
```

- ◆ Les commandes :

```
SET LOGSOURCE 'd:\temp'
RECOVER DATABASE
;
```

Si toutes les archives ne peuvent pas être ramenées d'un seul coup sur disque, il est possible de travailler série par série :

- ⇒ Récupérer la première série d'archives à un emplacement
- ⇒ Lancer le RECOVER en automatique en spécifiant l'emplacement



- ♦ Exemple de séquence pour une restauration à partir d'une source d'archives alternative.

```
SQL> SET LOGSOURCE 'd:\oracle\oradata\test\temp'
SQL> SET AUTORECOVERY ON
SQL> RECOVER DATABASE

ORA-00279: change 44629 generated at 08/07/2001 16:15:55 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00002.ARC
ORA-00280: change 44629 for thread 1 is in sequence #2

ORA-00279: change 45013 generated at 08/08/2001 06:59:49 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00003.ARC
ORA-00280: change 45013 for thread 1 is in sequence #3
ORA-00278: log file 'D:\ORACLE\ORADATA\TEST\TEMP\TEST00002.ARC' no longer needed for this
recovery

ORA-00279: change 45371 generated at 08/08/2001 07:00:01 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00004.ARC
ORA-00280: change 45371 for thread 1 is in sequence #4
ORA-00278: log file 'D:\ORACLE\ORADATA\TEST\TEMP\TEST00003.ARC' no longer needed for this
recovery

ORA-00279: change 45691 generated at 08/08/2001 07:00:15 needed for thread 1
ORA-00289: suggestion : D:\ORACLE\ORADATA\TEST\TEMP\TEST00005.ARC
ORA-00280: change 45691 for thread 1 is in sequence #5
ORA-00278: log file 'D:\ORACLE\ORADATA\TEST\TEMP\TEST00004.ARC' no longer needed for this
recovery

Log applied.
Media recovery complete.
```

