

M1 Informatique - Module Base de données

Premier Compte Rendu de TP



MARECHAL Anthony et Benoit

Encadrant : T. Grison

Table des matières

1	Introduction	4
2	Installation d'Oracle	4
2.1	Généralités	4
2.1.1	Définitions	4
2.1.2	Rôles	5
2.1.3	Délocalisation des bases de données	5
2.2	Préparation de l'installation	5
2.2.1	Modification des paramètres du noyau	5
2.2.2	Création des groupes et de l'utilisateur oracle	6
2.2.3	Les variables d'environnement de la base (oraenv.sh)	6
2.2.4	Création de l'arborescence	6
2.3	Lancement de l'installation	7
2.4	Installation d'Oracle	7
2.5	Connexion à la base de données et création de l'utilisateur	8
2.5.1	Connexion à la base de données	8
2.5.2	Création d'un utilisateur	9
3	Création d'une base de données	9
3.1	Préparation de l'environnement	9
3.2	Lancement de l'assistant et configuration des paramètres	10
4	Structure interne d'Oracle	11
4.1	Les tablespaces	11
4.1.1	Comparaisons des différents modes de tablespaces	11
4.1.2	Avantages et inconvénients des tablespaces local et dictionary	11
4.2	Etude des tablespaces spécifiques	11
4.2.1	Tablespace système	11
4.2.2	Tablespace d'annulation	11
4.2.3	Tablespace temporaire	12
4.3	Sauvegarde au travers des utilitaires Export/Import	12
4.3.1	Problématique	12
4.3.2	Options de la commande Export	13
4.3.3	Options de la commande Import	13
4.3.4	Fonctionnement des utilitaires	14
4.4	High Water Mark ou niveau de flottaison	14
4.4.1	Principe	14
4.4.2	Exemple	15
4.4.3	Fonctionnement	15
5	TP 2 : Index, ROWID et cluster	15
5.1	Les structures d'indexation	15
5.1.1	Les index primaire	16
5.1.2	Les index secondaires	18
5.1.3	Analyse et validité d'un index	18
5.1.4	Reconstruction d'un index	19
5.2	Organisation des tables	20
5.2.1	L'organisation indexée	20
5.2.2	Les ROWIDS	21
5.3	Les clusters	23
5.3.1	Jointure entre deux tables non liée par un cluster	23
5.3.2	Cluster par hashage	24
5.3.3	Cluster indexé	25

5.4	Optimal Flexible Architecture (OFA)	27
6	TP 3 : Dictionnaire de données, gestion de l'espace et sauvegardes	28
6.1	Le dictionnaire des données	28
6.2	Gestion de l'espace avec un tablespace en mode local	29
6.3	Gestion d'un tablespace en mode dictionary	30
6.4	Gestion de l'espace libre d'une table	31
7	Conclusion	33
8	Bibliographie	34

Table des figures

1	Notion de schéma.	4
2	Sélection du type d'installation.	7
3	Sélection de la configuration de base de données.	8
4	Spécification des options de la configuration de la base de données.	9
5	Schéma des utilitaires Export et Import.	12
6	Fonctionnement de la High Water Mark.	15

1 Introduction

Ce compte rendu a été réalisé dans le cadre de nos trois premiers travaux pratiques de base de données et a pour but de mettre en avant les concepts que l'on a étudié durant ces trois séances.

La méthode de travail que nous avons choisi de mettre en place a été de se répartir les travaux pratiques et la partie de compte rendu qui lui était associé en fonction des numéros de séances, l'un réalisant le TP numéro deux et l'autre le TP numéro trois.

Nous traiterons dans un premier temps de la mise en place et de l'installation d'Oracle sur Solaris, nous verrons ensuite comment on crée une base de données, puis nous étudierons la structure interne d'Oracle, suivit des Tp 2 et 3 qui traitent de l'optimisation des tables et de la gestion de l'espace.

2 Installation d'Oracle

2.1 Généralités

2.1.1 Définitions

Base de données : Une base de données, est un ensemble structuré et organisé permettant le stockage de grandes quantité d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données). Une base de données se traduit physiquement par un ensemble de fichiers sur le disque. Plus globalement, une base de données doit :

- Permettre l'accès aux données de façon simple (grâce à un langage de manipulation de données)
- Autoriser un accès aux informations à de multiples utilisateurs (partage, gestion de la concurrence)
- Pouvoir manipuler les données présentes dans la base de données (insertion, suppression, modification)
- Maintenir la base afin quel soit toujours cohérente et intégrée (avec élimination de la redondance par exemple)

Instance : Une instance comprend les processus et un espace mémoire sur le serveur (le System Global Area d'Oracle plus précisément) d'une base de données.

Schéma : Un schéma est un ensemble comprenant des structures de données et des données. Il appartient à un utilisateur de la base et porte le nom de ce dernier. Chaque utilisateur possède ainsi son propre schéma. Un schéma est donc une collection (ou un ensemble) nommé d'objets tels que des tables, vues, clusters, procédure et packages associé à un utilisateur précis. Quand un utilisateur de base de données est créé, son schéma est automatiquement créé. Un utilisateur ne pourra alors être associé qu'à un seul schéma et réciproquement. Nous pouvons voir Fig1 page 4, un schéma de cette notion.

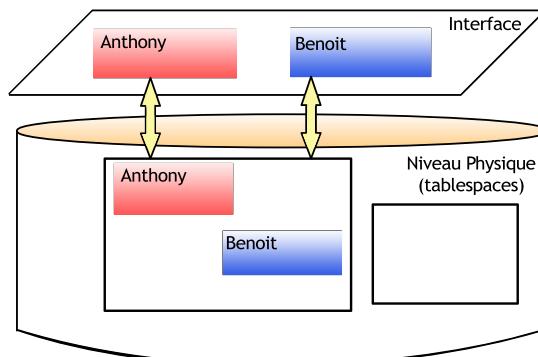


FIG. 1 – Notion de schéma.

Utilisateur : Un utilisateur de base de données correspond à un login qui aura recu certains privilèges / actions sur la base de données. Cet utilisateur sera stocké dans le dictionnaire de données et disposera d'un espace de stockage pour ses objets qui seront alors stockés dans son schéma. On peut donc assimiler un utilisateur avec son schéma.

2.1.2 Rôles

SYSDBA : L'utilisateur SYSDBA est en réalité une suite de privilèges qui comprend cinq opérations qu'un utilisateur Oracle peut avoir besoin de réaliser : démarrer / arrêter / créer / rétablir une base et faire le backup de celle-ci. A noter que la table v\$pwfile_users liste tous les utilisateurs ayant les privilèges de SYSDBA et qu'il est conseillé de réservrer ces privilèges à un administrateur d'une base uniquement.

SYS : L'utilisateur SYS possède toutes les tables et toutes les vues accessibles des utilisateurs du dictionnaire de données (contenant les informations de configuration d'Oracle). Aucun utilisateur Oracle ne peut modifier (mettre à jour, supprimer, ou insérer) des lignes contenues dans le schéma de SYS car toute activité pourrait ici compromettre l'intégrité des données de la base. Autrement dit, SYS est l'administrateur de la base physique, son schéma comprend la majeure partie des tables et vues du dictionnaire de données. Il est, en outre, le seul par défaut à pouvoir arrêter l'instance de la base.

SYSTEM : L'utilisateur SYSTEM est utilisé pour créer des tables et vues additionnelles afin de permettre l'affichage d'information administrative d'Oracle ou des informations sur les tables et vues internes d'Oracle spécifiées dans les options et outils de celui-ci. Il s'agit de l'administrateur de la base logique possédant moins de droits que l'utilisateur SYS.

SYSMAN : Il s'agit de l'administrateur par défaut de OEM (Oracle Entreprise Manager), plus précisément il s'agit du compte du super administrateur d'Entreprise Manager et peut par conséquent, créer et modifier les autres comptes administrateur de EM en plus de pouvoir administrer l'instance de la base.

DBSNMP : L'utilisateur DBSNMP est utilisé par Entreprise Manager (EM) pour contrôler la base de données. En effet EM utilise ce compte pour accéder aux statistiques de performance de la base. Cet utilisateur est créé/supprimé par catsnmp.sql. A noter que pour changer son mot de passe il est nécessaire d'arrêter le processus agentctl et modifier les entrées de ./network/admin/snmp_rw.ora

2.1.3 Délocalisation des bases de données

Les fichiers datafiles, c'est à dire les fichiers contenant les bases de données, sont placées dans le répertoire /opt/oracle/databases/ afin de pouvoir les récupérer en cas de problème avec Oracle : en délocalisant nos bases de données du répertoire par défaut nos améliorons donc la sécurité des données. Même si Oracle à un problème nos bases de données ne seront pas touchées.

2.2 Préparation de l'installation

2.2.1 Modification des paramètres du noyau

Selon les spécifications de la documentation d'installation d'Oracle il est nécessaire, de modifier les paramètres du noyau. Il faut pour cela se connecter en tant que root sur la machine, puis ajouter les lignes suivantes au fichier /etc/system.

Listing 1 – Modification des paramètres du noyau.

```
1 set shmsys:shminfo_shmmax=4294967295
2 set shmsys:shminfo_shmmin=1
3 set shmsys:shminfo_shmmni=100
```

```

4 set shmsys:shminfo_shmseg=10
5 set shmsys:shminfo_shmmni=100
6 set shmsys:shminfo_shmmns=2500
7 set shmsys:shminfo_shmmsl=256
8 set shmsys:shminfo_shmopm=100
9 set shmsys:shminfo_shmvmx=32767
10 set noexec_user_stack=1

```

2.2.2 Création des groupes et de l'utilisateur oracle

On distingue deux types d'utilisateurs d'une base de données :

- Les utilisateurs qui ont la possibilité d'ajouter des composants à la BD.
- Les administrateurs de la base de données et de son noyau, qui peuvent arrêter ou démarrer la base sont appelés DBA (Database Administrator).

Nous allons créer deux groupes sous Unix afin de séparer ces différents types d'utilisateurs :

```

groupadd -g 400 dba
groupadd -g 401 oinstall

```

Le groupe dba contiendra les administrateurs, alors que le groupe oinstall contiendra les utilisateurs de la base de données.

Créons maintenant l'utilisateur oracle, qui devra pouvoir ajouter de nouveaux composants à la base, tout en pouvant également la lancer et l'arrêter.

Pour cela nous allons définir cet utilisateur avec le groupe oinstall comme groupe principal, et le groupe dba en groupe secondaire.

```
useradd -u 400 -c "oracle" -m -d /export/home/oracle -g oinstall -G dba -s /usr/bin/ksh oracle
```

2.2.3 Les variables d'environnement de la base (oraenv.sh)

Dans le fichier oraenv.sh, la variable ORACLE_BASE désigne le répertoire racine d'Oracle, alors que la variable ORACLE_HOME désigne le répertoire de la version d'Oracle que l'on veut utiliser, ici il s'agit de la version 10r2. La variable ORACLE_SID désigne, quant à elle, la base de données sur laquelle on se connecte.

2.2.4 Création de l'arborescence

Nous allons maintenant créer l'arborescence des dossiers d'accueil d'Oracle. Comme nous allons créer ses dossiers sous le répertoire `/opt` et que nous allons utiliser la commande chown il est nécessaire d'être logué sous le compte root. Nous allons créer ses répertoires sous `/opt` car il s'agit du répertoire couramment utilisé lors de l'installation de logiciels commerciaux.

Création du répertoire racine

```
mkdir /opt/oracle
```

Création du répertoire des bases de données

```
mkdir /opt/oracle/databases
```

Création du répertoire accueillant les versions d'oracle

```
mkdir /opt/oracle/product
```

Création du répertoire accueillant la version que nous allons installer, à savoir la version 10r2

```
mkdir /opt/oracle/10r2
```

On désigne l'utilisateur oracle comme propriétaire des dossiers `/opt/oracle` et `/opt/oracle/databases`

```
chown -R oracle :oinstall /opt/oracle
```

```
chown -R oracle :dba /opt/oracle/databases
```

La préparation de l'installation est presque terminée, il faut maintenant redémarrer la machine afin que le noyau prenne en compte les modifications.

2.3 Lancement de l'installation

Avant de lancer l'installation il est nécessaire d'autoriser le serveur d'ouvrir une fenêtre graphique sur notre machine. Pour cela on utilise la commande `xhost + nom_machine`, en tant qu'utilisateur local : `xhost + MI105-06`. Ensuite en tant qu'utilisateur oracle, il faut exécuter les commandes suivantes :

```
# Lancement du script définissant les variables d'environnement d'oracle
./oraenv.sh

# export de l'affichage sur la machine
export DISPLAY=MI105-06 :0.0

# on se déplace ensuite dans le répertoire contenant l'installation d'oracle
/home1/cdoracle/Solaris/10r2/CD-db

# on lance l'install
./runInstaller
```

2.4 Installation d'Oracle

L'installation d'Oracle, se déroule par la succession d'écran que nous allons détailler ici :

- Sélectionner un type d'installation
 - > On choisit Enterprise Edition

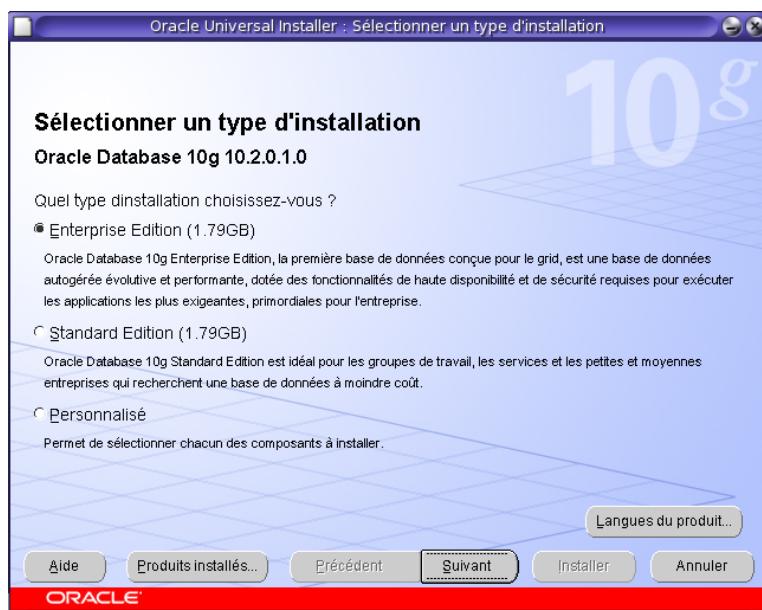


FIG. 2 – Sélection du type d'installation.

- Spécifier les détails du répertoire d'origine Oracle Home

-> Nom : `OraDb10g_home1`
-> Chemin : `/opt/oracle/production/10r2`

- Sélectionner une option de configuration

L'installeur permet de créer une première base de données nous choisissons donc
-> Créer une base de données

- Sélectionner une configuration de base de données
-> Usage général

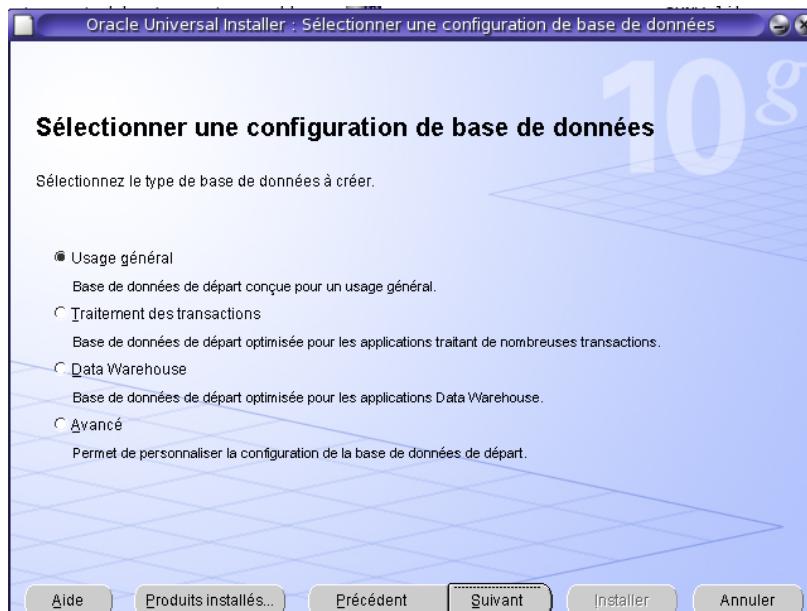


FIG. 3 – Sélection de la configuration de base de données.

- Spécifier les options de configuration de la base de données.

-> Nom : test
-> SID : test

- Spécifier l'option de stockage de base de données

-> Système de fichiers

- On spécifie le répertoire réservé aux bases :

/opt/oracle/databases/

- Indiquer les options de sauvegarde et de récupération

-> Ne pas activer les sauvegardes automatiques

- Indiquer les mots de passe de schéma de base de données.

On peu ici définir des mots de passes pour les différents comptes, SYS, SYSTEM, etc.

-> On utilisera le même mot de passe pour ces comptes.

Durant l'installation nous exécutons deux scripts en tant que root. Ces scripts réalisent des opérations sur le système afin d'accueillir Oracle.

Afin de vérifier la réussite de l'installation, nous pouvons regarder si les processus propre à oracle son lancé :

ps -ef | grep oracle

2.5 Connexion à la base de données et création de l'utilisateur

2.5.1 Connexion à la base de données

Pour nous loguer sous SQLPLUS depuis un compte étudiant il est nécessaire de réaliser les étapes suivantes :

- Ouvrir un terminal
- Prendre l'identité de l'utilisateur oracle : *su oracle*

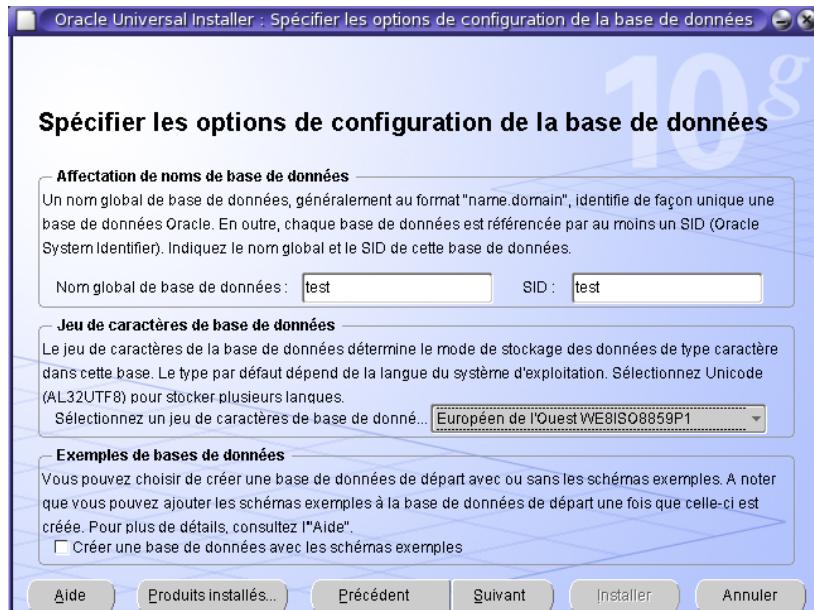


FIG. 4 – Spécification des options de la configuration de la base de données.

- Entrer le mot de passe correspondant : *iemoracle*
 - Fixer les variables d'environnement (en spécifiant, entre autres, la base de données que l'on souhaite lancer) :
`/export/home/oracle/oraenv.sh`
 - Lancer SQPLUS : `sqlplus /nolog`
 - Se connecter avec les privilèges SYSDBA : `connect / as sysdba`
 - Démarrer l'instance de la base de données : `startup`
- Et nous voici avec un prompt pour lancer les commandes SQL.

2.5.2 Crédation d'un utilisateur

Nous allons maintenant créer un utilisateur, pour cela on va utiliser l'instruction GRANT. L'utilisateur doit avoir un grand contrôle sur la base de données, mais il ne devra pas avoir de privilège d'administration. Pour cette raison nous allons lui attribuer les rôles connect et resource, mais pas le rôle dba.

```
CREATE USER benoit IDENTIFIED BY iembenoit;
GRANT connect, resource TO benoit;
```

3 Crédation d'une base de données

La création d'une base de données sous Oracle, est réalisée grâce à l'utilitaire DBCA. Il s'agit d'un assistant graphique, permettant de définir les différents paramètres nécessaire à la création de la base.

3.1 Préparation de l'environnement

Avant de lancer cet assistant, il faut vérifier que l'autorisation d'affichage est activée. On tape pour cela la commande suivante logué sous notre nom d'utilisateur : `xhost <nom_machine>`, où `<nom_machine>` est le nom de la machine obtenu avec la commande `hostname`. Puis sous l'utilisateur oracle on exporte l'affichage :

```
export DISPLAY=<nom_machine> :0.0
```

3.2 Lancement de l'assistant et configuration des paramètres

Pour lancer cet assistant, il faut être logué sous l'utilisateur Oracle puis lancer l'assistant par la commande :
`shell> dbca &`.

Avant de commencer une tel installation il est nécessaire de se poser des questions sur l'utilisation future de notre base. Si on ne la connaît pas vraiment, l'option Usage général conviendra. Si on la cerne assez bien, on peut utiliser une des deux options, Data Warehouse pour une base orienté application décisionnelle (améliore les recherches), et Traitement transactionnel, pour une base orienté application de saisie (beaucoup de modification sur la base).

Si vous connaissez avec précision l'utilisation que vous ferez de la base, l'option "Base de données personnalisée" est à préconiser.

Pour des raisons pédagogiques évidentes, nous choisirons cette dernière option.

Une succession d'écrans vont maintenant nous permettre de configurer notre nouvelle base de données, selon les paramètres suivants :

- **Etape 1** : Choix de l'action à réaliser, ici nous voulons créer une base de données. C'est donc cette option qu'il faut sélectionner.
- **Etape 2** : Sélection d'un modèle, nous allons personnaliser la base de données, nous ne choisirons donc pas de modèle. Choisissons l'option "Base de données personnalisée".
- **Etape 3** : Identification de la base de données. Cette étape permet de choisir le nom de la base de données et le SID de l'instance. Comme nous allons créer une base de données en mode dictionary nous appellerons la base "dbdico". Par soucis de simplicité, nous gardons le même nom pour le nom global et l'instance de la base.
- **Etape 4** : Option de gestion, choix entre une gestion centralisée ou locale.
- **Etape 5** : Choix des mots de passes des différents comptes. Cet écran permet de saisir les mots de passes de différents comptes de base d'Oracle.
- **Etape 6** : Option de stockage. Permet de choisir le système de fichiers.
- **Etape 7** : Emplacement des fichiers de la base de données. Nous choisissons maintenant l'emplacement des fichiers de la base. Nous les stockerons dans le répertoire `/opt/oracle/databases`. L'emplacement de ce répertoire est stratégique pour deux raisons :
 1. Son emplacement situé directement sous le répertoire `/opt/oracle` permet de partager les données avec les éventuelles autres versions d'Oracle contenues dans le répertoire `/opt/oracle/product`
 2. La séparation des fichiers contenant la base de données des autres fichiers d'Oracle, améliore la sécurité en cas de crash d'Oracle. Il peut également être intéressant de définir un répertoire monté sur un autre disque dur. Cela permettrait d'améliorer la rapidité des requêtes sur la base, et la sécurité en cas de crash du système et/ou d'Oracle.
- **Etape 8** : Configuration de la récupération. Cet écran permet d'indiquer l'emplacement et la taille de la zone de récupération.
- **Etape 9** : Contenu de la base de données. Cet écran permet de définir les composants à installer pour la base. Le choix de ces composants dépend de l'utilisation que l'on va faire de la base.
- **Etape 10** : Paramètres d'initialisation. Là encore, il est nécessaire de connaître à l'avance le type d'utilisation de la base de données, pour configurer au mieux les paramètres relatifs à la mémoire, le nombre maximum de processus, la taille du bloc, les jeux de caractères ainsi que le mode de serveur.
- **Etape 11** : Stockage de la base de données. On distingue deux types de gestion des blocs et de segments : le mode automatique et le mode de gestion au travers du dictionnaire des données. Nous choisirons le mode dictionary. Cela nous permettra par la suite de créer nos propres tablespaces en mode dictionary. Notons qu'une

base de données en mode dictionary, n'empêche pas la création de tablespace gérer de manière automatique (management local).

Il faut savoir qu'à moins de connaitre avec précision le type d'utilisation de la base de données, les paramètres par défaut d'Oracle permettent d'avoir une base pour un usage standard.

4 Structure interne d'Oracle

4.1 Les tablespaces

4.1.1 Comparaisons des différents modes de tablespaces

Un tablespace est un espace logique qui contient les objets stockés dans la base de données comme les tables ou les index, il est composé d'au moins un datafile, c'est à dire un fichier de données qui est physiquement présent sur le serveur à l'endroit stipulé lors de sa création. Chaque datafile est constitué de segments d'au moins un extent lui-même constitué d'au moins trois blocs : l'élément le plus petit d'une base de données. Or selon le mode de gestion d'un tablespace l'organisation de celui-ci ne sera pas la même. Le choix entre les deux modes de gestions possibles (local et dictionary) entraînera des différences dans la rapidité ou sur la quantité d'informations disponibles sur le tablespace d'où la nécessité d'étudier ces différentes gestions.

4.1.2 Avantages et inconvénients des tablespaces local et dictionary

Voici un tableau qui synthétise les avantages et inconvénients de ces deux modes :

	Avantages	Inconvénients
Mode Local	-Affranchissement de la gestion des extents -Evite les mésententes avec le dictionnaire de données -Gestion de l'espace du tablespace automatique	-Peu d'information sur les tablespaces
Mode dictionary	-Beaucoup d'informations sur les tablespaces enregistrés dans le dictionnaire de données	-Charge supplémentaire pour toutes opérations -Risque de fragmentation des fichiers -Ralentissement dû à des accès intempestifs au dictionnaire de données

4.2 Etude des tablespaces spécifiques

4.2.1 Tablespace système

Ce tablespace est le tablespace d'origine, créé automatiquement par Oracle et il contient le dictionnaire de données. Par défaut il existe toujours un tablespace baptisé SYSTEM qui contient le dictionnaire de données. Voici l'ordre de création d'un tablespace SYSTEM :

Listing 2 – Crédit d'un tablespace système.

```
1 CREATE TABLESPACE OUTILS
2   DATAFILE '/opt/oracle/databases/outil01.dbf'
3   SIZE 200M
4   DEFAULT STORAGE (INITIAL 512K NEXT 512K
5     PCTINCREASE 0 MAXEXTENTS 50);
```

4.2.2 Tablespace d'annulation

Le tablespace d'annulation (ou UNDO tablespace) est réservé à l'annulation des commandes du langage de définition de données (DDL) comme les insertions, mise à jour, etc.

Si nous supprimons, par exemple, certains tuples d'une table, alors le tablespace d'annulation contiendra les lignes ainsi supprimées avant d'indiquer les blocs à nouveaux libres. Ce mécanisme permet ainsi de garder la trace des données modifiées non validées, et c'est grâce à lui que la commande *Rollback* permet de restaurer des données pourtant altérées.

Voici l'ordre de création d'un tablespace d'annulation (contenant les rollbacks segments) :

Listing 3 – Création d'un tablespace d'annulation.

```
1 CREATE TABLESPACE RBS
2   DATAFILE '/opt/oracle/databases/rbs01.dbf'
3   SIZE 200M
4   DEFAULT STORAGE (INITIAL 1M NEXT 1M
5     PCTINCREASE 0 MAXEXTENTS 30);
```

4.2.3 Tablespace temporaire

Les tablespaces temporaires (TEMP ou temporary tablespaces) sont des tablespaces spécifiques aux opérations de tri, création d'index, opérations de jointures, etc. nécessitant un espace physique sur le disque du fait d'un manque de place dans le *sort area*.

Ce tablespace est réservé au système et n'est donc pas destiné à accueillir des objets de la base de données. Ils ont été créés par Oracle dès la version 9i car dans les versions précédentes ces données temporaires étaient gérées dans le tablespace standard se qui n'était pas aussi efficace qu'excentrer cette gestion dans des tablespaces spécifiques.

Voici l'ordre de création d'un tablespace temporaire :

Listing 4 – Création d'un tablespace temporaire.

```
1 CREATE TABLESPACE TEMP
2   DATAFILE '/opt/oracle/databases/temp01.dbf'
3   SIZE 200M
4   DEFAULT STORAGE (INITIAL 3M NEXT 3M
5     PCTINCREASE 0 MAXEXTENTS 30)
6   TEMPORARY;
```

4.3 Sauvegarde au travers des utilitaires Export/Import

4.3.1 Problématique

Comment sauvegarder une base de données et la restaurer ? Peut-on envoyer une base de données à quelqu'un ? Pouvons-nous restaurer une base de données sur un environnement différent de celui sur lequel on a extrait une base ?

La sauvegarde avec l'utilitaire Export est une méthode logique de sauvegarde, tout comme la restauration des données avec Import. Ces utilitaires vont nous permettre de sauvegarder le contenu d'une base de données dans un fichier *dump* spécifique à Oracle. Leurs avantages sont de pouvoir exporter une base sous un environnement différent de celui sur lequel on souhaite importer la base. Voici - Fig5 page 12 - un schéma simple du fonctionnement de ces utilitaires :

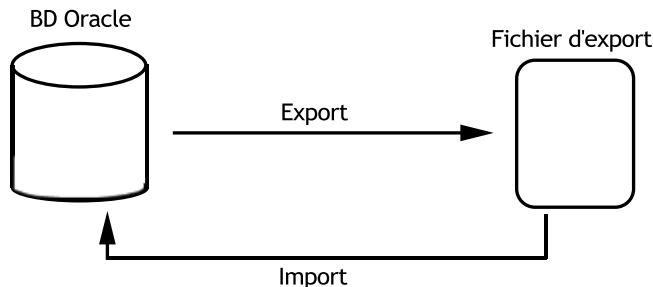


FIG. 5 – Schéma des utilitaires Export et Import.

4.3.2 Options de la commande Export

Afin de pouvoir utiliser la commande d'exportation voici un listing exhaustif de toutes les options de cette commande avec leurs descriptions et leurs valeurs par défaut.

Option	Description	Valeur par défaut
UserId	Chaîne de connexion à la base de données	
Buffer	Taille du buffer de transfert	4096
Compress	Compression des extents en un seul	Y
Constraints	Export des contraintes	Y
File	Nom du fichier DUMP	
Log	Nom du fichier de sortie du compte-rendu, pour voir les erreurs en particulier	expdat.dmp
Full	Export de toute la base	N
Grants	Export des privilèges	Y
Help	Affiche la liste des paramètres supportés, aucun DUMP généré	N
Indexes	Export des index	Y
Owner	Utilisateur à exporter	userid
Parfile	Fichier contenant les paramètres d'export	
Recordlength	Taille des enregistrements (migration SE)	
Record	Indique que l'export doit stocker les informations sur les exports et les objets exportés	Y
Inctype	Export incrémental (COMPLETE, CUMULATIVE, INCREMENTAL)	
Rows	Export des lignes	
Query	Définit une condition de filtre pour exporter un sous-ensemble	
Tables	Table(s) à exporter	
Consistent	Positionne sa session en READ ONLY le temps de l'export. Cela permet donc de préserver la cohérence des données exportés.	N
Direct	Chargement direct par tableau	N
Statistics	Analyse des objets exportés	ESTIMATE
Feedback	Affiche la progression de l'export tous les n enregistrements	N
Point_in_time_Recover	Indique si on autorise l'export des tablespaces	N
Recover_Tablespaces	Liste des tablespaces à sauvegarder	
Volsize	Nombre d'octets à écrire sur chaque volume bande	

4.3.3 Options de la commande Import

Comme demandé dans le sujet du compte rendu, voici maintenant le listing exhaustif des options de la commande Import.

Option	Description	Valeur par défaut
Userid	Chaîne de connexion à la base de données	
Buffer	Taille du buffer de transfert	10240
Commit	Ecriture régulière des blocs de données	N
File	Nom du fichier DUMP	expdat.dmp
Log	Nom du fichier de sortie du compte-rendu, pour voir les erreurs en particulier	
Fromuser	Utilisateur à exporter vers TOUSER	
Full	Import de tout le contenu du DUMP	N
Grants	Export des privilèges	Y
Help	Affiche la liste des paramètres supporté, aucun DUMP généré	N
Ignore	Ignore les erreurs	N
Indexes	Import des index	Y
Parfile	Fichier contenant les paramètres d'import	
Recordlength	Taille des enregistrements (migration SE)	
Rows	Import des données	Y
Destroy	Détruit les objets s'ils existent avant de les importer	N
Show	Liste le contenu du fichier d'export, aucune opération n'est effectuée dans la base.	N
Tables	Table(s) à importer	
Touser	Utilisateur destinataire	
Charset	Code alphabet du pays de référence s'il est différent de celui de la création de base	NLS_LANG
Point_in_time_recover	Indique si on autorise l'import des tablespaces	N
Skip_unusable_indexes	Permet de repousser la reconstruction de l'index après l'insertion des données dont ils dépendent	N
Analyze	Exécute la commande ANALYZE dans le fichier dump	Y
Feedback	Affiche la progression de l'import tous les n enregistrements	0
Volsize	Nombre d'octets dans le fichier pour chaque volume bande	

4.3.4 Fonctionnement des utilitaires

Voici un exemple de sauvegarde avec Export, nous sauvegardons ici l'intégralité de la base de données (l'option full étant mis à y), les données de la base seront également sauvegardés car l'option rows vaut par défaut y. Dans un prompt shell, avec la base de données arrêté, nous entrons :

```
exp userid=system/manager file=/opt/oracle/databases/export_full.dump log=/opt/oracle/databases/export_full.log
full=y
```

Et voici la commande Import à entrer pour restaurer ce que nous venons d'exporter :

```
imp userid=system/manager file=/opt/oracle/databases/export_full.dump log=/opt/oracle/databases/control/export_full.log
```

Ainsi nous avons sauvegardé notre base dans le fichier dump spécifique d'Oracle et restauré celui-ci simplement avec la commande Import.

4.4 High Water Mark ou niveau de flottaison

4.4.1 Principe

La notion d'High Water Mark appelé niveau de flottaison en français est un repère placé par Oracle pour indiquer jusqu'à quel bloc d'un extent les données ont été enregistrées. Ceci permet à Oracle de savoir quand il doit s'arrêter de parcourir le bloc lors du *fullscan* (balayage complet).

4.4.2 Exemple

Admettons que nous avons des blocs de 32 kilo-octects (la taille des blocs spécifiés à DB_BLOCK_SIZE vaut 32K). Si nous devions insérer 128 kilo-octects de données nous aurions la marque du High Water Mark sur le quatrième bloc ce qui signifie pour Oracle qu'il n'y a pas de données à lire au-delà de cette marque (les quatre premiers blocs de données étant utilisés). Si nous supprimons des données ou si nous mettons à jour les données avec des plus petites informations, la marque devrait redescendre or elle ne le fait pas, Oracle, lors des prochains *fullscan*, continuera à lire jusqu'à l'ancienne marque. Pour faire face à ce problème l'administrateur doit agir manuellement afin de réorganiser l'espace. Pour ce faire il peut réaliser un TRUNCATE ou bien utiliser les utilitaires Export et Import.

4.4.3 Fonctionnement

Pour mettre en avant le fonctionnement de cette notion d'High Water Mark (HWM) nous avons réalisé le schéma Fig6 page 15.

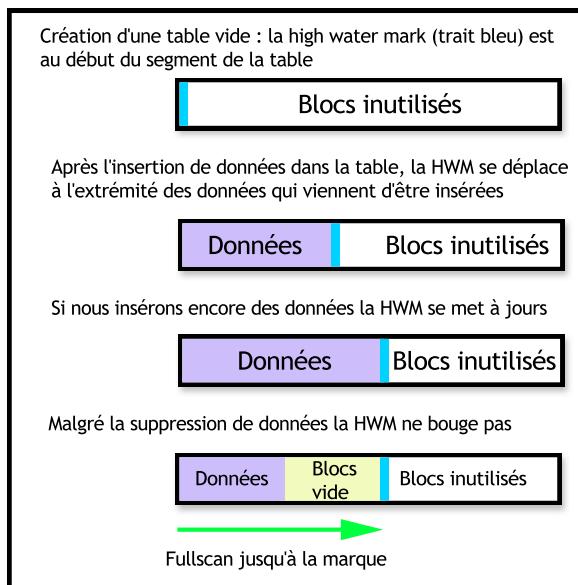


FIG. 6 – Fonctionnement de la High Water Mark.

La méthode de calcul pour avoir le pourcentage d'occupation inutile de la High Water Mark est assez simple :
$$HWM \% = (Niveau\ de\ HWM - Quantité\ de\ données\ réellement\ présente / Niveau\ de\ HWM) * 100$$

5 TP 2 : Index, ROWID et cluster

Dans cette partie nous étudierons les différents mécanismes d'optimisations d'une base de données. Nous verrons tout d'abord les structures d'indexations, suivi de l'organisation des tables notamment par l'analyse des ROWIDS et enfin nous étudierons le fonctionnement des clusters sous Oracle.

5.1 Les structures d'indexation

Les index sont des structures permettant d'améliorer considérablement les temps de recherches dans les tables. Aussi, lors de la création d'une table sous Oracle, on peut logiquement penser que ce dernier crée un index sur la clé primaire de la table, améliorant alors les requêtes portant sur la clé primaire d'une table. Afin de vérifier cette hypothèse nous allons créer une table, et effectuer des insertions. Puis nous testerons différentes requêtes de sélections et nous commenterons les résultats. Enfin nous les expliquerons à l'aide du dictionnaire de données.

5.1.1 Les index primaire

Nous allons ici créer un index primaire, insérer des données et effectuer des selections pour connaître son organisation.

```
1 ——creation
2 CREATE TABLE R1 (
3 i NUMBER(10) PRIMARY KEY,
4 j NUMBER(10),
5 k NUMBER(10)
6 );
7
8 ——insertions
9 INSERT INTO R1 VALUES(3 ,3 ,3);
10 INSERT INTO R1 VALUES(2 ,2 ,2);
11 INSERT INTO R1 VALUES(5 ,5 ,5);
12 INSERT INTO R1 VALUES(1 ,1 ,1);
13 INSERT INTO R1 VALUES(4 ,4 ,4);
14 INSERT INTO R1 VALUES(8 ,8 ,8);
15 INSERT INTO R1 VALUES(7 ,7 ,7);
16 INSERT INTO R1 VALUES(9 ,9 ,9);
17 INSERT INTO R1 VALUES(6 ,6 ,6);
18
19 —— selections
20 SELECT * FROM R1;
21
22      I          J          K
23 -----
24      3          3          3
25      2          2          2
26      5          5          5
27      1          1          1
28      4          4          4
29      8          8          8
30      7          7          7
31      9          9          9
32      6          6          6
```

Constat : Les résultats apparaissent dans l'ordre d'insertion.

```
1 SELECT i FROM R1;
2
3      I
4 -----
5      1
6      2
7      3
8      4
9      5
10     6
11     7
12     8
13     9
```

Constat : les résultats apparaissent triés.

```
1 SELECT j FROM R1;
2      J
3 -----
4      3
```

```
5      2
6      5
7      1
8      4
9      8
10     7
11     9
12     6
```

Constat : les résultats sont dans l'ordre d'insertion.

```
1 SELECT k FROM R1 ;
2
3      K
4 -----
5      3
6      2
7      5
8      1
9      4
10     8
11     7
12     9
13     6
```

Constat : Les résultats sont dans l'ordre d'insertion.

On constate que les résultats apparaissent dans l'ordre d'insertion sauf lorsque la requête porte seulement sur la clé primaire (la colonne i). Cela est parfaitement logique si un index à été créé sur cet attribut.

Afin de vérifier cette hypothèse, interrogeons le dictionnaire de données.

Comme nous allons utiliser pour la première fois le dictionnaire de données, profitons-en pour dégager une méthode de recherche d'information.

Etape 1 : Premièrement il faut trouver la table dans laquelle l'information cherchée doit se trouver. Pour cela nous affichons toutes les tables du dictionnaire ainsi que le commentaire associé en interrogeant la table DICTIONARY grâce à la requête suivante :

```
SELECT TABLE_NAME, COMMENTS FROM DICTIONARY;
```

Etape 2 : Une fois la table repérée, il reste à trouver quelles colonnes interroger, pour cela on utilise l'instruction *DESCRIBE <nom_table>* (ou *DESC <nom_table>*).

Remarque : on peut également demander directement l'affichage des tables avec leurs noms de colonnes grâce à la vue DICT_COLUMNS :

```
SELECT TABLE_NAME, COLUMN_NAME, COMMENTS FROM DICT_COLUMNS;
```

Après recherche dans le dictionnaire de données, il s'avère que les noms des index créés sous l'utilisateur courant, sont stockés dans la colonne INDEX_NAME de la table USER_INDEXES. D'où la requête

```
1 SELECT INDEX\_NAME FROM USER\_INDEXES ;
2
3 — On obtient le résultat suivant :
4
5 INDEX_NAME
6 -----
7 SYS_C005248
```

Un index à bien été créé automatiquement par le système, notre hypothèse est donc vérifiée : Oracle crée automatiquement un index sur la clé primaire d'une table.

5.1.2 Les index secondaires

Oracle, comme tout bon SGBD permet de définir à tout moment des index secondaires portant sur une ou plusieurs colonnes dans le but d'améliorer les recherches sur les attributs d'une table. Au niveau SQL, un index secondaire est créé selon la syntaxe :

CREATE INDEX <nom_index> ON <nom_table>(<liste_de_colonnes>);

Créons un index secondaire sur la colonnes j de notre table R1.

CREATE INDEX ind_r1_j on R1 (j);

Effectuons les mêmes requêtes de sélection que précédemment.

```
1 | SELECT * FROM R1;
2 |
3 |
4 |     I          J          K
5 |-----|-----|-----|
6 |     3          3          3
7 |     2          2          2
8 |     5          5          5
9 |     1          1          1
10 |    4          4          4
11 |    8          8          8
12 |    7          7          7
13 |    9          9          9
14 |    6          6          6
```

Constat : Les résultats apparaissent dans l'ordre d'insertion.

SELECT i FROM R1;

Constat : Les résultats apparaissent dans l'ordre trié.

SELECT j FROM R1;

Constat : Les résultats apparaissent dans l'ordre d'insertion.

SELECT k FROM R1;

Constat : Les résultats n'apparaissent pas trié pour la requête de sélection sur la colonne j, malgré l'existence de notre index sur cette colonne. Existence vérifier par la requête :

```
1 | SELECT INDEX_NAME ,UNIQUENESS FROM USER_INDEXES WHERE INDEX_NAME='IND_R1_J';
2 | INDEX_NAME
3 | -----
4 | IND_R1_J
```

On en conclut que index secondaire ne sont pas trié sous Oracle contrairement aux index primaire.

5.1.3 Analyse et validité d'un index

Oracle offre un mécanisme d'analyse des tables, index et cluster. Cette analyse permet entre autre de :

- Collecter et gérer des statistiques sur l'objet analysé
- Vérifier la validité de format de stockages
- D'identifier les lignes chaînées d'une table ou d'un cluster

La syntaxe SQL de l'analyse a la forme suivante :

ANALYZE <objet> <nom_objet> [VALIDATE STRUCTURE | LIST CHAINED ROWS];

Nous allons maintenant supprimer quelques lignes dans la table R1 puis nous vérifierons la validité de la structure de l'index créé automatiquement par Oracle sur la clé primaire :

```
1 | --- Destruction de quelques lignes
2 | DELETE FROM R1 WHERE i=1;
```

```

3 | DELETE FROM R1 WHERE i=3;
4 | DELETE FROM R1 WHERE i=5;
5 | DELETE FROM R1 WHERE i=6;
6 |
7 | -- Analyse de l'index de la cle primaire
8 | ANALYZE INDEX SYS\_\_C005143 VALIDATE STRUCTURE;
9 |
10| Index analyzed.

```

Aucune erreur n'est renvoyée par l'analyse, la structure est donc valide.

Différentes statistiques sur l'index sont maintenant disponibles dans la table INDEX_STATS, comme le nombre de blocs alloués, les lignes dernièrement supprimées et l'espace utilisé :

```

1 | SELECT BLOCKS, LF_ROWS, DEL_LF_ROWS, USED_SPACE, BTREE_SPACE FROM index_stats;
2 |
3 |   BLOCKS      LF_ROWS  DEL_LF_ROWS  USED_SPACE  BTREE_SPACE
4 |   -----  -----  -----  -----  -----
5 |     8          9          4        117       7996

```

On remarque notamment que l'index prend 8 blocs d'espace sur le disque, qu'il référence 9 lignes, et que 4 lignes ont récemment été supprimées.

Faisons de même pour l'index secondaire que nous avons créé :

```

1 | -- Analyse de l'index de la colonne j de R1
2 | ANALYZE INDEX IND_R1_J VALIDATE STRUCTURE;
3 |
4 | Index analyzed.
5 |
6 | SELECT BLOCKS, LF_ROWS, DEL_LF_ROWS, USED_SPACE, BTREE_SPACE FROM index_stats;
7 |
8 |   BLOCKS      LF_ROWS  DEL_LF_ROWS  USED_SPACE  BTREE_SPACE
9 |   -----  -----  -----  -----  -----
10|    8          9          4        126       7996

```

Les statutiques sur cet index sont les mêmes que pour l'index primaire. En effet, les index portant tous deux sur la même table et sur une colonne de même type ayant les mêmes valeurs, il est normal de retrouver des statistiques identiques.

5.1.4 Reconstruction d'un index

Reconstruisons maintenant un index, et relançons l'analyse :

```

1 | ALTER INDEX IND_R1_J REBUILD;
2 | ANALYZE INDEX IND_R1_J VALIDATE STRUCTURE;
3 | SELECT BLOCKS, LF_ROWS, DEL_LF_ROWS, USED_SPACE, BTREE_SPACE FROM index_stats;

```

Après analyse nous avons les nouvelles statistiques suivantes :

	BLOCKS	LF_ROWS	DEL_LF_ROWS	USED_SPACE	BTREE_SPACE
1	8	6	0	84	7996

On remarque deux informations essentielles grâce aux statistiques :

1. Les lignes qui avaient été supprimées ne sont plus référencées par l'index. L'index a en quelque sorte été "nettoyé".
2. L'espace utilisé a été diminué.

La reconstruction des index est donc très importante et doit être réalisée régulièrement lorsqu'il s'agit d'index portant sur des tables subissant beaucoup de modifications. Comme les tables d'une application de saisie par exemple.

5.2 Organisation des tables

5.2.1 L'organisation indexée

Oracle permet de définir ce qu'il appelle une "organisation indexée" sur une table. Nous allons dans cette partie voir de quoi il s'agit. Pour cela nous allons créer une table avec ce type d'organisation, nous effectuerons différentes requêtes SQL afin d'obtenir un maximum d'informations sur celles-ci.

Création de la table R2 en "organisation indexée" :

```
1 CREATE TABLE R2 (
2   i NUMBER(10) PRIMARY KEY,
3   j NUMBER(10),
4   k NUMBER(10)
5 )
6 ORGANIZATION INDEX;
7
8 — Insertion des même valeurs que pour la tables R1.
9
10 INSERT INTO R2 VALUES(3 ,3 ,3);
11 INSERT INTO R2 VALUES(2 ,2 ,2);
12 INSERT INTO R2 VALUES(5 ,5 ,5);
13 INSERT INTO R2 VALUES(1 ,1 ,1);
14 INSERT INTO R2 VALUES(4 ,4 ,4);
15 INSERT INTO R2 VALUES(8 ,8 ,8);
16 INSERT INTO R2 VALUES(7 ,7 ,7);
17 INSERT INTO R2 VALUES(9 ,9 ,9);
18 INSERT INTO R2 VALUES(6 ,6 ,6);
```

Réalisons les mêmes requêtes de sélection que sur la table R1 :

```
1 SELECT * FROM R2;
2   I          J          K
3   —————— —————— ——————
4   1          1          1
5   2          2          2
6   3          3          3
7   4          4          4
8   5          5          5
9   6          6          6
10  7          7          7
11  8          8          8
12  9          9          9
```

Constat : Les résultats apparaissent triés.

SELECT i FROM R2;

Constat : Les résultats apparaissent triés.

SELECT j FROM R2;

Constat : Les résultats apparaissent triés.

SELECT k FROM R2;

Constat : Les résultats apparaissent triés.

On constate que les résultats affichés sont toujours triés, peu importe la colonne sur laquelle porte la requête. En fait, si les valeurs insérées dans les colonnes j et k n'étaient pas les mêmes que pour la colonne i, nous aurions des résultats toujours triés selon i, donc nous n'aurions pas de résultats triés pour les requêtes portant sur j et k.

Un tel résultat ne peut s'expliquer que par l'existence d'un index plaçant sur la clé primaire de la table. Rappelons qu'un index plaçant est un index dont les clés sont directement insérées triées. L'avantage notable d'un tel index est de réaliser une recherche dichotomique améliorant alors considérablement les temps de réponses des requêtes.

Oracle permet également de définir des index sur des fonctions. Comme par exemple des fonctions mathématiques ou des fonctions sur les chaînes de caractères. En voici quelques exemples.

Exemple 1 : Création d'un index sur des fonctions mathématiques

```
1 CREATE INDEX IND_R2_k ON R2 (SIN(k)*COS(k));
2
3 Index cree.
```

Exemple 2 : Création d'un index sur les chaînes de caractères.

Pour prouver qu'il est possible de créer un index sur les chaînes de caractères. Créons une table contenant une colonne en VARCHAR2, insérons quelques valeurs, et tentons de créer l'index.

```
1 CREATE TABLE TChaine (
2 idTChaine NUMBER PRIMARY KEY,
3 chaine varchar2(50)
4 );
5
6 INSERT INTO TChaine VALUES(1 , 'message 1');
7 INSERT INTO TChaine VALUES(2 , 'bonjours ');
8 INSERT INTO TChaine VALUES(3 , 'fleurs ');
9 INSERT INTO TChaine VALUES(4 , 'animal');
```

Créons un index sur la longueur des chaînes.

```
1 CREATE INDEX IND_TCHAINE_ch ON TChaine (LENGTH(chaine));
2
3 Index cree.
```

Oracle a bien accepté la création de l'index avec la fonction LENGTH, on peut conclure qu'il accepte la création d'index sur n'importe quelles fonctions.

5.2.2 Les ROWIDS

Chaque ROWID est unique pour une ligne. Ils décrivent l'emplacement physique de l'enregistrement notamment par un numéro de fichier qui identifie le datafile, un numéro de block de l'enregistrement (BLOCK_NUMBER) et un numéro de ligne dans le block (ROW_NUMBER). Depuis Oracle 8 les ROWIDS permettent également de connaître le segment et l'objet auquel l'enregistrement appartient¹.

L'analyse des ROWDID d'une table est donc un très bon moyen d'appréhender l'organisation physique des enregistrements.

Observons les ROWIDS de la table R1 et de la table R2.

```
1 SQL> select ROWID FROM R1;
2
3 ROWID
4 -----
5 AAAMfbAAEAAAAAWAAD
6 AAAMfbAAEAAAAAWAAB
7 AAAMfbAAEAAAAAWAAA
8 AAAMfbAAEAAAAAWAAE
9 AAAMfbAAEAAAAAWAAC
10 AAAAfbaAEAAAAAWAAI
```

¹<http://www.orafaq.com/glossary/faqglosr.htm>

```

11 | AAAMfbAAEAAAAAWAAG
12 | AAAMfbAAEAAAAWAAF
13 | AAAMfbAAEAAAAWAAH
14 |
15 | SQL> select ROWID FROM R2;
16 |
17 | ROWID
18 |
19 | *BAEAAAawCwQL+
20 | *BAEAAAawCwQP+
21 | *BAEAAAawCwQT+
22 | *BAEAAAawCwQX+
23 | *BAEAAAawCwQb+
24 | *BAEAAAawCwQf+
25 | *BAEAAAawCwQj+
26 | *BAEAAAawCwQn+
27 | *BAEAAAawCwQr+

```

L'affichage des ROWIDS sans traitement n'est pas très parlant, d'autant plus qu'ils sont codés sur une base peut commune : la base 64. Afin de retirer des informations de ces ROWID on utilise le package PL/SQL DBMS_NUMBER. D'où la requête :

```

1 | SELECT DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(
2 |   ROWID) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
3 |   ROWID_ROW_NUMBER(ROWID) as row_num FROM R1;
4 |
5 | BLOCK_NUM      OBJECT      FNO      ROW_NUM
6 | -----|-----|-----|-----|
7 | 22      51163      4      3
8 | 22      51163      4      1
9 | 22      51163      4      0
10 | 22      51163      4      4
11 | 22      51163      4      2
12 | 22      51163      4      8
13 | 22      51163      4      6
14 | 22      51163      4      5
15 | 22      51163      4      7

```

Les informations sur les ROWIDS sont maintenant beaucoup plus parlantes. On peut constater que toutes les lignes de la table R1 ont été stockées dans le même bloc (22) d'un même objet (51163) d'un même fichier (4). Et donc que chaque enregistrement est stocké dans une ligne différente du bloc (de 0 à 8).

Demandons maintenant l'affichage des informations des ROWIDS de la table R2.

```

1 | SELECT DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(ROWID
2 |   ) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
3 |   ROWID_ROW_NUMBER(ROWID) as row_num FROM R2
4 |
5 | ERREUR a la ligne 1 :
6 | ORA-06553: PLS-306: numero ou types d'arguments errones dans appel a
7 | 'ROWID_ROW_NUMBER'

```

La requête pose problème. En effet, les ROWIDS des tables en organisation indexées (ORGANIZATION INDEX) sont des ROWIDs logiques constitué de la valeur de la clé primaire et d'un identifiant de bloc.

Et comme on peut le constater dans la documentation officielle² Le package DBMS_ROWID ne permet pas de lire les ROWIDS logique (appelé UROWID, pour "Universal ROWID" dans les documentations).

²http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96612/d_rowid.htm

5.3 Les clusters

Les clusters permettent de rapprocher physiquement les lignes de tables différentes afin d'améliorer leur jointure. Pour cela, le critère de rapprochement (la clé de cluster) doit être la même que le critère de jointure. Afin de mettre en avant les avantages des clusters dans des cas concrets, nous allons procéder en trois étapes :

- Premièrement, nous créerons deux tables liées entre elles par une clé étrangère, et nous analyserons (grâce aux informations des ROWIDS) le nombre de blocs qu'il est nécessaire de lire pour effectuer une jointure.
- Deuxièmement, nous créerons un cluster par hachage entre deux tables, et nous effectuerons la même jointure que précédemment. Nous pourrons alors comparer le nombre de blocs lu dans les deux cas.
- Troisièmement, nous créerons un cluster indexé entre deux tables, nous comparerons la aussi le nombre de blocs lu.

5.3.1 Jointure entre deux tables non liée par un cluster

```
1 — Creation de la table R3(i)
2
3 CREATE TABLE R3 (
4   i NUMBER PRIMARY KEY);
5
6 — Insertion de valeurs
7
8 INSERT INTO R3 VALUES(1);
9 INSERT INTO R3 VALUES(2);
10 INSERT INTO R3 VALUES(3);
11 INSERT INTO R3 VALUES(4);
12
13 — Creation de la table R4(i,j)
14
15 CREATE TABLE R4 (
16   i NUMBER,
17   j NUMBER,
18   CONSTRAINT fk_r4_j FOREIGN KEY (j) REFERENCES R3(i)
19 );
20
21 — Insertion de valeurs
22
23 INSERT INTO R4 VALUES(1,2);
24 INSERT INTO R4 VALUES(2,3);
25 INSERT INTO R4 VALUES(3,3);
26 INSERT INTO R4 VALUES(4,4);
27
28
29 — Analyse des ROWIDS :
30
31 SELECT i, DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num, DBMS_ROWID.ROWID_OBJECT(
32   ROWID) as object, DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno, DBMS_ROWID.
33   ROWID_ROW_NUMBER(ROWID) as row_num FROM R3;
34
35   I  BLOCK_NUM    OBJECT      FNO    ROW_NUM
36   -- -----
37   1    78        51170       4      0
38   2    78        51170       4      1
39   3    78        51170       4      2
40   4    78        51170       4      3
41
42 SELECT j, DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num, DBMS_ROWID.ROWID_OBJECT(
43   ROWID) as object, DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno, DBMS_ROWID.
```

```

42      ROWID_ROW_NUMBER(ROWID) as row_num FROM R4;
43
44      J    BLOCK_NUM     OBJECT      FNO      ROW_NUM
45      2      94      51172      4      0
46      3      94      51172      4      1
47      3      94      51172      4      2
48      4      94      51172      4      3

```

On constate que les enregistrements de la table R3 et de la table R4 ne sont pas dans le même block, ni dans le même objet. Notons qu'Oracle aurait peut-être pu deviner que l'on souhaite un rapprochement entre R3 et R4 car R4 possède une clé étrangère sur R3, ce qui sous entend que des jointures auront probablement lieu. On constate également que tous les enregistrements d'une même table sont dans le même bloc et que chaque ROWID possède un numéro différent.

Dans notre exemple, si on exécute une requête SQL réalisant une jointure entre les tables R3 et R4 avec $j=3$ tel que :

```

1   SELECT R3.i , R4.j FROM R3 , R4
2   WHERE R3.i = R4.j — jointure
3   AND     R4.j = 3;

```

Cela demandera la lecture du bloc 78 et du bloc 94. Soit la lecture de deux blocs. Il serait plus intéressant de ranger les mêmes valeurs dans les mêmes blocs, ce qui limiterait le nombre de bloc lu pour une telle requête.

5.3.2 Cluster par hashage

Les hash-cluster sont adapté à des tables supportant de nombreux accès ponctuels et peu de mises à jour. Ce type de cluster associe une liste de blocs contenant toutes les valeurs renvoyées par la fonction hash-code pour cette clé. Ainsi lorsqu'on veut réaliser la jointure entre deux tables, les valeurs communes entre les mêmes tables se retrouve dans les mêmes blocs, ont limitera ainsi considérablement le nombre de blocs lu pour effectuer une jointure.

Créons maintenant un cluster par hachage entre deux tables de même structure et contenant les mêmes valeurs que les tables créées précédemment. Nous poserons ce cluster sur la colonne i de la première table, et sur la colonne j de la seconde table, afin d'améliorer les jointures entre ces deux tables.

```

1   — Creation d'un cluster par hachage
2   CREATE CLUSTER hash(i NUMBER(10)) HASHKEYS 100;
3
4
5   — Creation de la table R3_hash(i)
6
7   CREATE TABLE R3_hash
8   ( i NUMBER(10) NOT NULL)
9   CLUSTER hash (i);

```

Un cluster doit obligatoirement avoir une colonne non nulle (NOT NULL), ici il s'agit de la colonne i, car une clé primaire est implicitement non nulle.

```

1   — Insertion de valeurs
2
3   INSERT INTO R3_hash VALUES(1);
4   INSERT INTO R3_hash VALUES(2);
5   INSERT INTO R3_hash VALUES(3);
6   INSERT INTO R3_hash VALUES(4);
7
8   — Creation de la table R4_hash(i,j)
9
10  CREATE TABLE R4_hash (
11    i NUMBER(10),
12    j NUMBER(10) NOT NULL
13 ) CLUSTER hash (j);

```

```

14
15 — Insertion de valeurs
16
17 INSERT INTO R4_hash VALUES(1 ,2);
18 INSERT INTO R4_hash VALUES(2 ,3);
19 INSERT INTO R4_hash VALUES(3 ,3);
20 INSERT INTO R4_hash VALUES(4 ,4);

```

Analysons maintenant les ROWIDS de ces deux tables

```

1 SELECT i , DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(
2   ROWID) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
3   ROWID_ROW_NUMBER(ROWID) as row_num FROM R3_hash;
4
5   I   BLOCK_NUM   OBJECT      FNO   ROW_NUM
6   ---  ---  ---  ---  ---
7   3     118    51173        4      0  <-- bloc 118
8   2     155    51173        4      0
9   4     188    51173        4      0
10  1     192    51173        4      0
11 SQL>
12 SELECT j , DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(
13   ROWID) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
14   ROWID_ROW_NUMBER(ROWID) as row_num FROM R4_hash;
15
16   J   BLOCK_NUM   OBJECT      FNO   ROW_NUM
17   ---  ---  ---  ---  ---
18   3     118    51173        4      0  <-- bloc 118
19   3     118    51173        4      1  <-- bloc 118
20   2     155    51173        4      0

```

On constate que les enregistrements de ces deux tables ont été placés dans le même objet (51173) et que les valeurs similaires ont été rangées dans les mêmes blocs. En effet pour I=3 de la table R3_hash et pour J=3 de la table R4_hash. On remarque que ces valeurs ont été placées dans le même bloc, ici 118.

Donc si on exécute une requête SQL réalisant une jointure entre les tables R3 et R4 avec j=3 tel que :

```

1 SELECT R3_hash .i , R4.j FROM R3_hash , R4_hash
2 WHERE R3_hash .i = R4_hash .j
3 AND     R4_hash .j = 3;

```

Nous aurons à lire seulement le bloc 118, puisque la valeur 3 des colonnes i et j est stockée dans le même bloc. Le cluster par hachage offre donc un avantage évident pour le traitement des jointures.

5.3.3 Cluster indexé

Les clusters indexés sont adaptés aux tables supportant de nombreux accès séquentiels par intervalle de valeurs de clé. Les valeurs de la clé du cluster sont stockées dans un index. A chaque valeur de l'index est associée la liste des blocs qui contiennent l'ensemble des lignes des tables du cluster dont la clé correspond à la valeur. Dernière particularité, la valeur de la clé n'est stockée qu'une seule fois dans chaque bloc.

On peut alors faire l'hypothèse que pour des valeurs proches le numéro des blocs associés doit également être proche.

Pour mettre en place un cluster indexé il faut procéder en deux étapes :

- Créer le cluster
- Créer un index sur le cluster

Reste alors à associer au cluster les colonnes des tables dont on veut accélérer la jointure :

```

1 — Creation d'un cluster
2 CREATE CLUSTER cidx(i NUMBER(10)) ;
3
4 — Creation de l'index sur le cluster
5 CREATE INDEX IND_cidx ON CLUSTER cidx ;
6
7
8 — Creation de la table R3_chash(i)
9
10 CREATE TABLE R3_cidx
11 ( i NUMBER(10) NOT NULL)
12 CLUSTER cidx (i);
13
14 — Creation de la table R4_chash(i,j)
15 CREATE TABLE R4_cidx (
16 i NUMBER(10) ,
17 j NUMBER(10) NOT NULL
18 ) CLUSTER cidx (j);
19
20 — Insertion de valeurs proches (successives)
21 INSERT INTO R3_cidx VALUES(1);
22 INSERT INTO R3_cidx VALUES(2);
23 INSERT INTO R3_cidx VALUES(3);
24 INSERT INTO R3_cidx VALUES(4);
25
26 INSERT INTO R4_cidx VALUES(1,2);
27 INSERT INTO R4_cidx VALUES(2,3);
28 INSERT INTO R4_cidx VALUES(3,3);
29 INSERT INTO R4_cidx VALUES(4,4);
30
31 — Analysons les ROWIDS :
32
33 SELECT i , DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(
ROWID) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
ROWID_ROW_NUMBER(ROWID) as row_num FROM R3_cidx ;
34
35      I    BLOCK_NUM     OBJECT      FNO      ROW_NUM
36 -----
37      4        212    51176          4          0
38      1        214    51176          4          0
39      2        215    51176          4          0
40      3        216    51176          4          0 <-- bloc 216
41
42 SELECT j , DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) as block_num , DBMS_ROWID.ROWID_OBJECT(
ROWID) as object , DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) as fno , DBMS_ROWID.
ROWID_ROW_NUMBER(ROWID) as row_num FROM R4_cidx ;
43
44      J    BLOCK_NUM     OBJECT      FNO      ROW_NUM
45 -----
46      4        212    51176          4          0
47      2        215    51176          4          0
48      3        216    51176          4          0 <-- bloc 216
49      3        216    51176          4          1 <-- bloc 216

```

On retrouve ici le même principe que pour le CLUSTER par hachage, à savoir le placement des mêmes valeurs dans les mêmes blocs. Mais ce n'est pas tout, en regardant de plus près les numéros de blocs on s'aperçoit que ces derniers sont quasiment contigu :

Valeur 2 : bloc 215

Valeur 3 : bloc 216

Valeur 4 : bloc 212

Ce qui n'était pas du tout le cas pour les blocs du cluster par hachage :

Valeur 2 : 155

Valeur 3 : 118

Valeur 4 : 188

Nous pouvons en conclure que le cluster indexé à un avantage supplémentaire sur le cluster par hachage en accélérant les jointures sur des plages de valeurs, ce qui confirme notre hypothèse.

5.4 Optimal Flexible Architecture (OFA)

Tout comme les répertoires du système de fichier d'un système d'exploitation, Oracle doit également structurer correctement ces répertoires. Oracle recommande ainsi d'utiliser la structure OFA qui est un ensemble de règles d'installation et de configuration qui permet la création de base Oracle plus performante (plus rapide, plus fiables, plus faciles à faire évoluer et nécessitant peu de maintenance). Nous allons détailler les règles de cette architecture.

Nommer les répertoires :

/point_de_montage/nom_reperoire_OFA/nom_proprietaire_du_reperoire

Exemple : */u01/app/oracle*, correspond au répertoire d'installation du logiciel Oracle, propriété de l'utilisateur Oracle.

Règles pour permettre l'installation et le fonctionnement simultané de plusieurs versions d'Oracle :

/point_de_montage/nom_reperoire_OFA/nom_proprietaire/product/num_version_d_oracle

Exemple : */u01/app/oracle/product/8.0.5*

Nommer les fichiers d'administration des bases :

OFA propose de créer un jeu de répertoires concernant les fichiers de configuration, de trace, de log pour chaque base ayant pour chemin d'accès :

-reperoire_accueil_oracle/admin/nom_de_la_base/indication du type de fichier de config

Avec comme type de fichier de configuration :

adhoc (script SQL utilisé), arch (stockage des fichiers Redo-log archivés), adump (fichier d'audit si l'option est activée, bdump (trace des processus de la base), cdump (fichier d'erreur du noyau Oracle), create (fichier de création de la base), exp (fichier d'export de la base), logbook (fichier contenant l'historique des actions effectuées sur la base, et pfile (fichier d'initialisation de la base).

Exemple pour le fichier d'initialisation de la base PROD, init-PROD.ora : */u01/app/oracle/admin/PROD/pfile*

Nommer les fichiers des bases :

OFA préconise de ne pas mélanger les fichiers de notre bases de données avec ceux des logiciels Oracle, nous avons ainsi les règles suivantes :

- pour les fichiers de contrôle : */point_de_montage/oradata/nom_de_la_bd/control.ctl*
- pour les fichiers Redo-log : */point_de_montage/oradata/nom_de_la_bd/redon.rdo*
- pour les fichiers de données : */point_de_montage/oradata/nom_de_la_bd/xy.dbf*

Avec *oradata* le répertoire racine hébergeant les fichiers de données de toutes les bases ; *x* le nom du tablespace auquel est rattaché le fichier et *y* le numéro de fichier sur deux chiffres.

Exemple pour le premier fichier du tablespace SYSTEM de la base PROD : */u01/oradata/PROD/system01.dbf*.

Synthèse :

L'organisation des fichiers proposés par OFA rend donc l'administration de la base plus simple. Voici quelques

exemples mettant en avant les biens fait de cette architecture :

- `/*app/oracle/product/*` : permet de lister les répertoires d'installation des différentes versions d'Oracle
- `/*app/oracle/admin/*` : permet de lister tout les fichiers de configuration des bases de données Oracle
- `/*oradata/TEST/*` : permet de lister les fichiers de la base de données TEST

Il est donc évident que le respect de tels règles est nécessaire pour la gestion du système de fichier lié aux bases dont ont a la gestion.

6 TP 3 : Dictionnaire de données, gestion de l'espace et sauvegardes

Dans cette partie nous travaillerons sur la partie du dictionnaire de données se rapportant aux tablespaces, fichiers et segments. On étudiera entre autres la gestion des tablespaces en mode LOCAL et en mode DICTIONARY à travers différentes expériences.

6.1 Le dictionnaire des données

Le dictionnaire de données est le référentiel qui décrit tous les objets physiques et logiques d'une base de données Oracle. Il s'agit d'un ensemble de tables et de vues systèmes contenant les informations relatives à la structure de la base de données.

Avant de commencer les manipulations, assurons nous que le tablespace SYSTEM est bien mode DICTIONARY car si ce n'est pas le cas il ne sera pas possible de créer des tablespaces dans ce mode.

Pour cela nous allons interroger la table `dba_tablespaces` et les colonnes `tablespace_name` et `extent_management` qui donne le type de gestion du tablespace :

```
1 | SELECT tablespace_name , extent_management FROM dba_tablespaces WHERE tablespace_name='
2 | SYSTEM';
3 |
4 | TABLESPACE_NAME          EXTENT_MAN
5 | _____
6 | SYSTEM                  DICTIONARY
```

Le tablespace est bien en mode DICTIONARY.

Nous pouvons généraliser la requête pour obtenir le mode de gestion de tous les tablespaces avec la requête suivante :

```
1 | SELECT tablespace_name , extent_management FROM dba_tablespaces;
2 |
3 | TABLESPACE_NAME          EXTENT_MAN
4 | _____
5 | SYSTEM                  DICTIONARY
6 | UNDOTBS1                LOCAL
7 | SYSAUX                  LOCAL
8 | TEMP                     LOCAL
9 | USERS                   LOCAL
```

Toujours en interrogeant le dictionnaire de données on peut obtenir les tablespaces de tries et les tablespaces par défaut. Il faut pour cela interroger la table `dba_tablespaces` et la colonnes `allocation_type`.

Les tablespace de tries sont des tablespace temporaire leur type d'allocation est UNIFORM, donc pour connaître il faut taper la requête :

```
1 | SELECT tablespace_name , allocation_type FROM dba_tablespaces
2 | WHERE allocation_type ='UNIFORM';
3 |
4 | TABLESPACE_NAME          ALLOCATIO
```

5	
6	TEMP UNIFORM

Pour les tablespaces par défaut leur type d'allocation SYSTEM. On obtient donc leur liste par la requête :

```

1 SELECT tablespace_name , allocation_type FROM dba_tablespaces
2 WHERE allocation_type = 'SYSTEM';
3
4 TABLESPACE_NAME          ALLOCATIO
5
6 UNDOTBS1                 SYSTEM
7 SYSAUX                    SYSTEM
8 USERS                     SYSTEM

```

- Déterminons maintenant les caractéristiques d'extension des fichiers des tablespaces :

```

1 SELECT tablespace_name as tablespace , initial_extent as init , next_extent as next ,
      min_extents as min , max_extents as max , pct_increase as inc , min_extlen as min
   FROM dba_tablespaces ;

```

Cette requête donne la politique de gestion des extents de tous les tablespaces. Il peut être plus intéressant de demander cette information pour un tablespace donnée, en précisant une clause WHERE :

```

1 SELECT tablespace_name as tablespace , initial_extent as init , next_extent as next ,
      min_extents as min , max_extents as max , pct_increase as inc ,
2 FROM dba_tablespaces
3 WHERE tablespace_name=<nom_tablespace>;

```

6.2 Gestion de l'espace avec un tablespace en mode local

Les tablespaces en mode local sont ceux créés par défaut par Oracle depuis la version 9i. Dans ce mode Oracle gère tout seul la politique l'allocation de l'espace. Cette partie consiste à appréhender plus précisément, quelle est cette politique. Pour cela nous allons :

- Créer un tablespace en mode LOCAL
- Créer une table et l'ajouter à ce tablespace
- Effectuer des insertions tout en regardant l'évolution du fichier, des segments et des extents du tablespace

1 - Crédation d'un tablespace en mode LOCAL avec fichier autoextensible et de taille maximale fixée.

```

1 CREATE TABLESPACE TSLOCAL1
2 DATAFILE '/opt/oracle/databases/tslocal_f1.dbf' SIZE 100K
3 AUTOEXTEND ON
4 NEXT 10K
5 MAXSIZE 500K
6 EXTENT MANAGEMENT LOCAL;

```

2 - Crédation d'une table associée à ce tablespace

```

1 CREATE TABLE table_tslocal (
2 etudiant VARCHAR2(100)
3 )
4 TABLESPACE TSLOCAL1;

```

3 - Insertions de lignes, statistiques et évolution de l'espace

Nous allons maintenant insérer des lignes dans la table et regarder l'évolution du fichier */opt/oracle/databases/ts-local_f1.dbf*, du nombre d'extensions et du détail d'un segment. Toutes ces informations seront obtenues par les instructions suivantes :

- Evolution du fichier */opt/oracle/databases* :

```
1 ls -ld fichier
```

Cette commande est à taper dans le Shell, sous l'utilisateur oracle. Elle donne la taille du fichier donné en paramètre.

- Nombres d'extensions alloués au tablespace :

```
1 SELECT sum(extents) FROM dba_segments  
2 WHERE tablespace_name=nom_tablespace ;
```

Cette instruction SQL effectue la somme du nombre d'extensions de chaque segment dont le tablespace est nom_tablespace.

- Détails sur les segments :

```
1 SELECT sg.tablespace_name , segment_name , extents , blocks , bytes FROM dba tablespaces ts  
     , dba_segments sg  
2 WHERE ts.tablespace_name = sg.tablespace_name  
3 AND ts.tablespace_name = nom_tablespace ;
```

Cette instruction donne diverses informations sur l'ensemble des segments d'un tablespace.

Espace libre du tablespace : *SELECT * FROM dba_free_space WHERE tablespace_name=nom_tablespace* ;

Affiche diverses informations sur l'espace libre d'un tablespace.

Par soucis de clarté et de concision, nous ne mettrons pas le listing des insertions ni le détail des requêtes permettant d'obtenir les informations recherchées. Nous avons préféré synthétiser ces données dans le tableau suivant :

Nb lignes insérées	Nb extents	Taille fichier dbf
0	1	114688
16	1	114688
256	1	114688
512	1	114688
1024	1	114688
2048	2	180224
4096	3	245760
8192	6	442368

Remarque : Les insertions ont été effectuées selon la procédure suivante :

INSERT INTO table_tslocal VALUES ('abcdefgij');

Puis nous avons utilisé l'instruction suivante pour remplir la base de manière exponentielle :

*INSERT INTO table_tslocal SELECT * FROM table_tslocal;*

On remarque qu'à chaque fois qu'un extent est ajouté au segment, le poids du fichier dbf augmente de 65536 octets.

6.3 Gestion d'un tablespace en mode dictionary

Procérons de la même manière pour un tablespace géré en mode dictionary.

- Création du tablespace

```
1 CREATE TABLESPACE TSDIC
2 DATAFILE '/opt/oracle/databases/tsdico.dbf' SIZE 100K
3 EXTENT MANAGEMENT DICTIONARY
4 DEFAULT STORAGE (
5     INITIAL 10K
6     NEXT 10K
7     MINEXTENTS 1
8     MAXEXTENTS 3
9 );
```

On vérifie que notre tablespace est bien en mode dictionary :

```
1 SELECT tablespace_name , extent_management FROM dba_tablespaces WHERE tablespace_name='
TSDIC';
2
3 TABLESPACE_NAME          EXTENT_MAN
4 -----
5 TSDIC                   DICTIONARY
```

- Création d'une table associé à ce tablespace

```
1 CREATE TABLE table_ts1 (
2 etudiant VARCHAR2(100)
3 )
4 TABLESPACE TSDIC;
5
6 Table creee.
```

- Insertions de lignes, statistiques et évolution de l'espace

Nb lignes insérées	Nb extents	Taille fichier dbf	Espace libre	Blocs libre
0	1	114688	81920	10
16	1	114688	81920	10
256	1	114688	81920	10
512	1	114688	81920	10
1024	1	114688	81920	10
2048	3	114688	49152	6

On constate qu'en mode dictionary la taille du fichier n'augmente pas en fonction du nombre d'extents, contrairement au mode local.

Arriver à 2048 lignes, nous ne pouvons plus insérer de nouvel enregistrement, ce qui est logique puisque nous avons défini MAXEXTENTS à 3, et qu'il faut 6 extents pour stocker 2048 lignes.

Nous préconisons d'utiliser le mode local, qui est le mode par défaut de gestion de tablespace par Oracle depuis la version 9i. Car dans ce mode l'administrateur de la base est certain que si beaucoup d'insertions sont réalisés le fichier database (dbf) sera étendu proportionnellement au nombre de d'extents nouvellement alloué au système.

6.4 Gestion de l'espace libre d'une table

Afin de mieux comprendre la gestion de l'espace libre d'une table, nous avons réalisé les tests suivants :

```
1 --- Creation d'une table .
2
3 CREATE TABLE maTable (
```

```
4      id NUMBER(3) ,
5      client VARCHAR2(50)
6 );
7
8 — Analyse d'une table
9 ANALYZE TABLE maTable COMPUTE STATISTICS ;
10 select * from dba_tables where table_name='MA_TABLE';
11
12 INSERT INTO MATABLE VALUES (1, 'chaine0');
13 INSERT INTO MATABLE VALUES (2, 'chaine1');
14 INSERT INTO MATABLE VALUES (3, 'chaine2');
15 INSERT INTO MATABLE VALUES (4, 'chaine3');
16 INSERT INTO MATABLE VALUES (5, 'chaine4');
17 INSERT INTO MATABLE VALUES (6, 'chaine5');
18
19 — Analyse de la table
20 ANALYZE TABLE maTable COMPUTE STATISTICS ;
21 select * from dba_tables where table_name='MA_TABLE';
22
23 DELETE FROM MATABLE WHERE id > 4;
24
25 — Analyse de la table
26 ANALYZE TABLE maTable COMPUTE STATISTICS ;
27 select * from dba_tables where table_name='MA_TABLE';
```

7 Conclusion

Nous avons eu, tout au long de la réalisation des TP, de petits problèmes. La plupart de ceux-ci ont toujours pu être résolus grâce au fascicule rouge et à la documentation officielle. Cependant lors d'un problème réseau de la salle A105 nous avons été déconnecté de notre ordinateur et lors du reboot, la base de données créée en mode dictionary à malheureusement été altérée. Aucune des différentes tentatives que nous avons essayées n'ont fonctionnées ce qui nous a obligé de procéder à une réinstallation de cette base ce qui nous a fait perdre du temps. Malgré tout les TP ce sont en général très bien passé.

Nous avons, jusque là toujours eu uniquement des cours ou aspect théorique sur le fonctionnement interne d'Oracle. Avec ce premier compte rendu nous avons vu, dans la pratique, le fonctionnement d'Oracle ce qui est toujours beaucoup plus parlant. La mise en place d'Oracle, la création de base de données et différents tests sur la gestion interne d'Oracle, ceci sous un environnement Solaris, nous a permis de rencontrer un nouvel aspect des bases de données, complémentaire à ce que nous avions appris l'année dernière et qui touche le fonctionnement interne d'un système de gestion de données. Nous avons donc pu réaliser l'intérêt qu'il y a à connaître le fonctionnement interne d'Oracle et ne pas se contenter de maîtriser uniquement le langage SQL. Aujourd'hui, connaître le fonctionnement du système de gestion de données nous paraît donc nécessaire à la création d'une bonne base de données. Avec toutes les fonctionnalités que nous venons d'étudier, Oracle nous apparaît comme un très bon système de gestion de base de données.

8 Bibliographie

Ce rapport réalisé à partir des ouvrages [1, 2, 3], des sites Internet [4, 5, 6, 7, 8, 9], de nos cours, travaux dirigés et pratique de licence [10] et de la documentation officielle d'Oracle [11].

Références

- [1] Christian Soutou. SQL pour Oracle. Edition Eyrolles. ISBN 2-212-11410-9.
- [2] Gilles Briard. Oracle 8i sous Linux. Edition Eyrolles. ISBN 2-212-09135-4.
- [3] Alain Moizeau. Oracle 8 Administration. Edition ENI. ISBN 2-7460-1037-2.
- [4] Didier Deleglise. <http://didier.deleglise.free.fr/dba/>.
- [5] Rédacteurs developpez.com. <http://jaouad.developpez.com/> et <http://orafrance.developpez.com/>.
- [6] Section Architecture Oracle de developpez.com. <http://oracle.developpez.com/guide/architecture/>.
- [7] Ecole supérieur SUPINFO. <http://www.supinfo.com>.
- [8] OraFAQ. <http://orafaq.com/articles>.
- [9] Serveur pédagogique de l'université. <http://ufrsciencestech.u-bourgogne.fr>.
- [10] Notion d'high water mark. <http://www.tafora.fr/div/highwatermark.doc.html>
- [11] Documentations officielles d'Oracle. <http://ufrsciencestech.u-bourgogne.fr/mil-tc2/docs>.