Partie 3

10. Sauvegarde et restauration.

- Solutions de sauvegarde et récupération
- Stratégies de sauvegarde
- Utilitaire RMAN

11. Transfert et chargement de données

- Les différents modes d'export et d'import
- Sqlloader et chargement de données
- Transfert de données Oracle-MySQL

12. Optimisation SQL

- Génération d'un plan d'exécution
- Éléments d'optimisation : Index, vues matérialisées
- Optimiser les performances de l’instance

Références

- Oracle 10g Administration, Olivier Heurtel, 2005, 489p.
- Administration Oracle 10G, Partie I, G. Mopolo-Moké, MBDS / UNSA NICE 2005/ 2006
- Oracle11g Administration, Razvan Bizoï, Eyrolles, 2011, ISBN : 978-2-212-12898-7
- Oracle11g Sauvegarde et sécurité, Razvan Bizoï, Eyrolles, 2011, ISBN : 978-2-212-12899-4
- Oracle 11g New Features for Administrators Summary Sheets, Version 1.2 Editor: Ahmed Baraka
- RECOVERY MANAGER G. Mopolo-Moké prof. MBDS UNSA 2005/ 200

10.Sauvegarde et Restauration de la base

- La sauvegarde et la restauration sont des processus critiques pour assurer la disponibilité et l'intégrité des données dans une base de données Oracle.
- La sauvegarde permet de créer une copie des fichiers de la base de données, y compris les données, les structures de stockage, les paramètres du système, les journaux de transactions, etc.
- La restauration est utilisée lorsque des données sont perdues, corrompues ou supprimées de la base de données. Elle consiste à récupérer les fichiers de sauvegarde créés lors de la sauvegarde et à les ramener à leur emplacement d'origine.
- La sauvegarde obéit à une stratégie :
  - Quoi sauvegarder : totalité, tablespace, uniquement les données sensibles, etc.
  - Quand : fréquence pluri quotidienne, quotidienne, hebdomadaire, etc.
  - Comment : à froid, à chaud, physiquement, logiquement

- Il y a deux types de sauvegardes :

  - La sauvegarde logique consiste à enregistrer des données de la base sur un fichier externe.
  - Oracle fournit les deux utilitaires : Export et Import
    - L'export peut servir pour faire une sauvegarde complète d'une base de données, la sauvegarde est dite FULL
    - A l'import, il faudra préparer la base cible avec les mêmes tablespaces, datafiles pour pouvoir recevoir les données.

  - La sauvegarde physique consiste en une copie de l'ensemble des fichiers composant la base de données :
    - Fichiers de contrôle, les controlfiles
    - Fichier d'initialisation, le pfile ou le spfile
    - Fichiers de données, les datafiles
    - Les journaux de transactions, les redologs

- La sauvegarde peut se faire à **froid** c'est à dire **base arrêtée** ou à **chaud** (uniquement avec le mode **ArchiveLog**).

- Depuis la 9i, Oracle propose l'utilitaire **RMAN**, pour Recovery MANager, afin de gérer les sauvegardes et les restaurations des bases de données.

  - Il permet d'effectuer des sauvegardes complètes ou partielles. On parle couramment de backup.

  - Les sauvegardes complètes, dites full, sont des sauvegardes de l'ensemble des blocs de la base.

  - Les sauvegardes partielles, dites incrémentielles, sont des sauvegardes des blocs modifiés depuis une précédente sauvegarde full ou incrémentielle.

- Une stratégie de sauvegarde possible est de faire une sauvegarde full à froid le week-end (ce qui permet en plus aux DBA d'avoir un créneau horaire pour faire des modifications de paramétrages internes) et des sauvegardes incrémentielles à chaud en semaine.

10.1. Stratégies de sauvegarde

10.1.1. Backup à froid, manuel, NOARCHIVELOG

– Ce type de backup sauvegarde la base de données, base arrêtée.

Les étapes de sauvegarde en mode **sans archive (save_BDOracle.sql)**

```
SQL> SHUTDOWN IMMEDIATE   -- Fermeture de la base de données pour avoir des fichiers synchronisés
```

```
                                        -- save_BDOracle.sql
-- Variables d'environnement de SQL*Plus de formatage de l'affichage
Set feedback off
Set Linesize 200
Set Heading off
Set Pagesize 0
Set Trimspool off
Set Verify off
define repertoire ='D:\BD_ORCL_SAVE' --  répertoire de destination des fichiers sauvegardés
define fichier_control=control_backup.sql  -- définition du fichier de sortie

spool &fichier_control

select 'host copy  ' || name || ' &repertoire ' from v$datafile order by 1 ;
select 'host copy  ' || member || ' &repertoire ' from v$logfile order by 1 ;
select 'host copy  ' || name  || ' &repertoire ' from v$controlfile order by 1 ;
select 'host copy  ' || name  || ' &repertoire ' from v$tempfile order by 1 ;
spool off

@&fichier_control
```

```
SQL>STARTUP
```

10.1.2. Backup à froid, manuel, ARCHIVELOG

- Ce type de backup sauvegarde la base de données, base arrêtée.
- La base elle-même mise en **archivelog**

- Modes de sauvegarde :
  - Backup complet à la création ou à intervalles réguliers (suivre les mêmes étapes qu'en mode noarchivelog)
  - Backup partiel de la base
  - Archivage automatique ou manuelle des fichiers REDO LOG
  - Backup du fichier de contrôle en cas de modification de la structure de la base (ajout d'un tablespace ...)

Backup partiel :

1. Backup partiel d'un Tablespace OFFLINE en 4 étapes

Etape 1 : Identifier les fichiers appartenant au tablespace X

```
sql>SELECT file_name FROM dba_data_files
        WHERE tablespace_name = 'X';
```

Etape 2 : Mettre le Tablespace X OFFLINE NORMAL

```
sql>ALTER TABLESPACE X OFFLINE NORMAL;
```

Etape 3 : Utiliser les commandes de l'OS (cp, tar, dump, ...) pour sauvegarder effectivement les fichiers sur bande ou disque

Etape 4 : Réactiver le Tablespace X

```
sql>ALTER TABLESPACE X ONLINE;
```

2. Backup d'un Tablespace (TS) ONLINE en 4 étapes

Etape 1 : Identifier les fichiers appartenant au tablespace X

```
sql>SELECT file_name FROM dba_data_files
WHERE tablespace_name = 'X';
```

Etape 2 : Indiquer à Oracle le début de la sauvegarde de X

```
sql>ALTER TABLESPACE X BEGIN BACKUP ;
```

Etape 3 : Utiliser les commandes de l'OS (cp, tar, dump, ...) pour sauvegarder effectivement les fichiers sur bande ou disque

Etape 4 : Informer Oracle de la fin de la sauvegarde du tablespace X

```
sql>ALTER TABLESPACE X END BACKUP ;
```

Note : La sauvegarde de plusieurs TS peut se faire séquentiellement ou en parallèle
BEGIN BACKUP X BEGIN BACKUP Y - sauvegardes –
END BACKUP X END BACKUP Y.

3. **Sauvegarde du fichier de contrôle**

1. Sauvegarde du fichier de contrôle Base fermée. Utiliser les commandes de l'OS pour sauvegarder

2. Sauvegarde du fichier de contrôle Base Ouverte

```
sql>ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

Génère le texte de la commande de création du fichier de contrôle

ou

```
sql>ALTER DATABASE BACKUP CONTROLFILE nomfich [REUSE] ;
```

Génère une copie du fichier de contrôle

10.1.3. Backup à chaud, manuel, ou RMAN (archivelog obligatoire)

– Ce type de backup sauvegarde la base de données, base ouverte.

– La base elle-même mise en **archivelog**

Savechaud_Oracle.sql

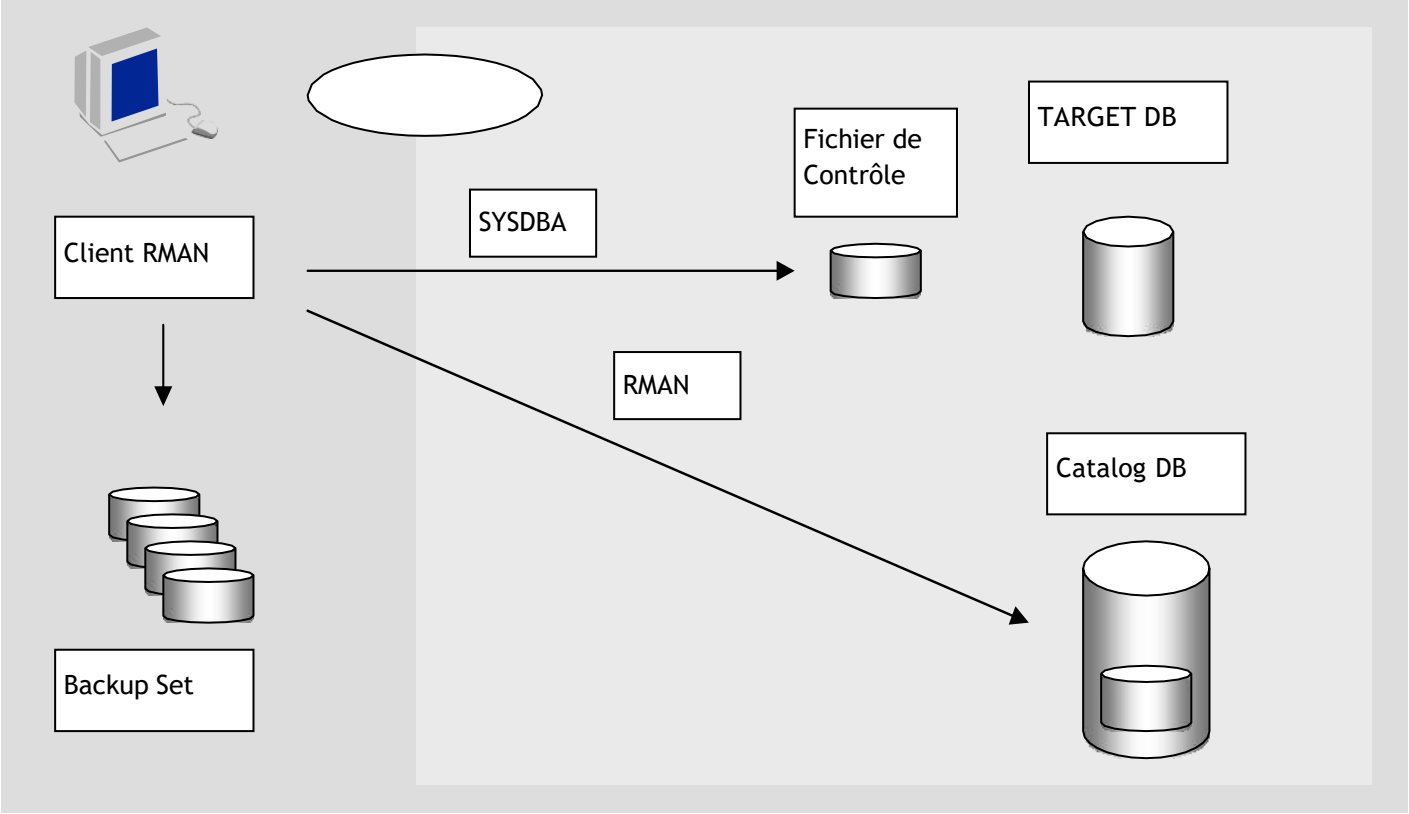
```
SET feedback off pagesize 0 heading off verify off linesize 100 trimspool on
PROMPT veuillez entrer le chemin du répertoire destinataire des sauvegardes
ACCEPT repertoire
PROMPT veuillez entrer le chemin du premier fichier
ACCEPT fichier
PROMPT veuillez entrer le chemin du second fichier
ACCEPT spool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

Le script précédant doit générer et exécuter un autre script de la forme :

```
spool c:\oracle\sauvegarde\hot_backup.sql ;
archive log list ;
alter system switch logfile ;
 alter tablespace SYSTEM begin backup  ;
 alter tablespace RBS begin backup  ;
 alter tablespace USERS begin backup  ;
 alter tablespace TEMP begin backup  ;
 alter tablespace TOOLS begin backup  ;
 alter tablespace INDX begin backup  ;
 alter tablespace DRSYS begin backup  ;
 host copy C:\ORACLE\ORADATA\BD0\USERS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\DR01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\TOOLS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\INDX01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\RBS01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\TEMP01.DBF c:\oracle\sauvegarde
 host copy C:\ORACLE\ORADATA\BD0\SYSTEM01.DBF c:\oracle\sauvegarde
 alter tablespace SYSTEM end backup  ;
 alter tablespace RBS end backup  ;
 alter tablespace USERS end backup  ;
 alter tablespace TEMP end backup  ;
 alter tablespace TOOLS end backup  ;
 alter tablespace INDX end backup  ;
 alter tablespace DRSYS end backup  ;
alter database backup controlfile to 'c:\oracle\sauvegarde\control.ctl' REUSE ;
alter system switch logfile ;
archive log list ;
SPOOL off ;
```

10.1.4. Utilitaire RMAN (Recovery MANager)

Rman est un utilitaire (exécutable) livré avec toute version Oracle.
Il gère les backups en utilisant un référentiel stocké dans une base Oracle ou dans le fichier de contrôle de la base sauvegardée.



– RMAN permet donc les types de sauvegardes suivants :

- De type **COMPLET** (ou FULL) : on sauvegarde tous les blocs ;
- De type **DIFFÉRENTIEL :** on sauvegarde uniquement les blocs modifiés depuis la précédente sauvegarde de niveau n ou inférieur ;
- De type **CUMULATIF :** on sauvegarde uniquement les blocs modifiés depuis la précédente sauvegarde de niveau n-1.

– RMAN utilise des niveaux de sauvegarde (3) :

- Niveau 0 : Sauvegarde de référence : l'ensemble des blocs contenant des données
- Niveau 1 : Sauvegarde tous les blocs modifiés depuis la plus récente sauvegarde incrémentale de niveau 0.
- Niveau 2 : sauvegarde tous les blocs modifiés depuis la plus récente sauvegarde incrémentale de niveau 0, 1 ou 2

10.1.4.1. Les commandes utiles de RMAN

CONFIGURE	permet de paramétrer le fonctionnement de RMAN
BACKUP	Sauvegarde les DB Files, archive logs, backups et copies
CHANGE	Change le statut d'un backup ou d'une copie dans le catalogue RMAN
LIST	affiche des infos générales sur l'état des sauvegardes/restaurations
COPY	Fait des copies 'images' des database files, archived logs, backups
CROSSCHECK	Vérifie l'existence des backup pièces, copies proxy et disques copies
DELETE	Efface les backups et copies, passe les enregistrements de méta données du CONTROL FILE au statut DELETED ou supprime les méta données du CATALOG
RECOVER	Fait un media recovery sur les backups et copies RMAN
REPORT	Fait un rapport sur diverses infos utiles : quels fichiers ont besoin de backups, sont obsolètes ou irrécupérables, liste les fichiers de la DB
RESTORE	Restaure les backups et copies RMAN
RUN	Utile pour exécuter certaines commandes (sans les entrer directement) comme ALLOCATE CHANNEL ou SET configure la session RMAN courante
SHOW	affiche les commandes CONFIGURE courantes

10.1.4.2. Connexion à RMAN

On peut lancer RMAN et se connecter à son référentiel de différentes façons :

- Lancement de RMAN sans connexion au référentiel

```
C:>RMAN
```

- Connexion à une cible locale (on utilise le CONTROL FILE comme référentiel) et l'environnement ORACLE_SID)

```
RMAN> connect TARGET sys/password@targetSID
```

- On se connecte à un référentiel externe (le catalogue)

```
RMAN> connect CATALOG rman_user/password@rmanSID
```

- On se connecte à la cible par défaut en authent externe

```
C:>RMAN target /
```

10.1.4.3. Gestion des sauvegardes

10.1.4.3.1. Lister les sauvegardes disponibles

- Sauvegardes de la base complète :

```
RMAN> list backup;
```

- Sauvegardes d'un fichier particulier :

```
RMAN> list backup of datafile 'chemin_datafile';
```

- Sauvegardes d'un tablespace particulier :

```
RMAN> list backup of tablespace TBS_NAME;
```

- Lister les fichiers pour lesquels il manque des sauvegardes si on veut répondre à la politique de sauvegarde définie (durée de rétention, nombre de sauvegardes...)

```
RMAN> report need backup;
```

- Lister les sauvegardes obsolètes (en dehors de la fenêtre de sauvegarde définie)

```
RMAN> report obsolete;
```

- Vérifier les sauvegardes effectuées par RMAN et encore physiquement présentes dans la zone de sauvegarde :

```
RMAN> crosscheck backup;
```

- Supprimer les sauvegardes obsolètes :

```
RMAN> delete obsolete;
```

10.1.4.3.2. Sauvegarder la totalité d'une base

- Si le mode CONTROLFILE AUTOBACKUP est réglé sur ON, une copie du fichier de contrôle ainsi que du SPFILE sera effectuée après la sauvegarde.

```
RMAN> configure CONTROLFILE AUTOBACKUP ON;
```

- Backup complet de la base par défaut

```
RMAN> BACKUP DATABASE;
```

- Sauvegarde la base de données ainsi que les fichier d'archive (REDOLOG journalisés).

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

10.1.4.3.3. Sauvegarder un tablespace

```
RMAN> BACKUP TABLESPACE system, users, tools, undo;
```

10.1.4.3.4. Sauvegarder un fichier de données

```
RMAN> BACKUP DATAFILE '?/oradata/bdtest/users01.dbf',
                      ?/oradata/bdtest/system.dbf';
```

- On peut également stipuler le numéro d'id du datafile à sauvegarder :

```
SQL> select file_id,file_name from dba_data_files;
  FILE_ID FILE_NAME
  ------- ----------------------------------------------
    1     .../oradata/ bdtest /system01.dbf
    2     .../oradata/ bdtest /undotbs01.dbf
    3     .../oradata/bdtest/users01.dbf
    4     .../oradata/ bdtest /example01.dbf
```

- Sauvegarde donc le fichier ".../oradata/ bdtest /example01.dbf" :

```
RMAN> BACKUP DATAFILE 4;
```

10.1.4.3.5. Sauvegarder le fichier de contrôle

```
RMAN> BACKUP CURRENT CONTROLFILE ;
```

Si on est en mode AUTOBACKUP ON, une copie du fichier sera effectuée à chaque fin de sauvegarde.

10.1.4.3.6. Sauvegarder les journaux de transactions (redo-log)

- On ne sauvegarde JAMAIS les REDO-LOG courant avec RMAN
- L'utilisation de RMAN nécessite que la base cible archive ses redo-log (mode ARCHIVELOG).

```
RMAN> BACKUP ARCHIVELOG ALL;
```

- Sauvegarde les journaux de transactions archivés du dernier mois écoulé :

```
RMAN> BACKUP ARCHIVELOG
TIME BETWEEN 'SYSDATE-31' AND 'SYSDATE';
```

- **Autres paramètres de la sauvegarde**

Paramètre	Exemple	Explication du fonctionnement
FORMAT	FORMAT '/tmp/%U'	Spécifie une localisation et un nom pour les pièces de sauvegardes.
FILESPERSET	FILESPERSET 20	Limite le nombre de fichiers de données ou de log archivés par jeu de sauvegarde
MAXSETSIZE	MAXSETSIZE 5G	Spécifie la taille maximum d'un jeu de sauvegarde
COPIES	COPIES 2	Spécifie le nombre de copies de chaque jeu de sauvegarde
TAG	TAG 'backup_du_lundi'	Spécifie une étiquette pour la sauvegarde. Par défaut RMAN, génère cette étiquette

Exemples :

RMAN> BACKUP TABLESPACE tools, indx, undotbs FORMAT '?/oradata/%U';

RMAN> BACKUP FILESPERSET 20 FORMAT='AL_%d/%t/%s/%p' ARCHIVELOG LIKE '%arc_dest%';

10.1.4.3.7. Sauvegarde incrémentale

- Lors d'un backup incrémental, on doit toujours spécifier le niveau 0 qui correspond à une sauvegarde complète de la base.

```
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

- Ensuite, on peut créer des sauvegardes à un niveau plus élevé.

```
RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

Dans cet exemple, seuls les blocs ayant été modifiés depuis la sauvegarde de niveau 0 seront sauvegardés.

10.1.4.3.8. Validation des sauvegardes

- On peut tester le fonctionnement d'une sauvegarde sans générer de sortie avec l'option VALIDATE.

```
RMAN>BACKUP VALIDATE DATABASE ARCHIVELOG ALL;
RMAN>BACKUP VALIDATE TABLESPACE tools;
```

- Cependant, une fois une sauvegarde effectuée on peut tester que la restauration se fera sans encombre. La commande à exécuter est donc la suivante :

```
RMAN> RESTORE DATABASE VALIDATE ;
```

10.1.4.3.9. Copie d'un fichier

- L'outil RMAN permet également de réaliser des copies de fichiers.
- Ces copies sont équivalentes à des copies de fichiers OS.
- Cela peut s'avérer nécessaire lorsque l'on désire réaliser des restaurations « user-managed ».

```
RMAN> COPY CURRENT CONTROLFILE TO '/save/btest_rman/cf_testcopie.ctl';
RMAN> COPY DATAFILE 1 TO '/save/btest_rman/dbf1_bkp.dbf'
```

10.1.4.4. Restuaration de la base

10.1.4.4.1. Récupération complète de la base de données

- Pour récupérer la base de données complète, il faut la monter simplement. Si elle est encore démarrée dans un état instable, il faut tenter de l'arrêter : d'abord par un SHUTDOWN IMMEDIATE, et si ça ne répond pas, par un SHUTDOWN ABORT.

```
RMAN> startup force mount;
RMAN> recover database;
RMAN> alter database open;
```

- Si la récupération échoue, la base de données a peut-être besoin d'une restauration complète. Dans ce cas, il faut préalablement restaurer les fichiers à partir d'une sauvegarde avant de les récupérer.

10.1.4.4.2. Restauration complète de la base de données

- Pour restaurer la base de données complète, il faut la monter simplement. Si elle est encore démarrée dans un état instable, il faut tenter de l'arrêter : d'abord par un SHUTDOWN IMMEDIATE, et si ça ne répond pas, par un SHUTDOWN ABORT.

NB 1 : RMAN refusera d'écraser les fichiers de la base de données s'ils existent déjà, et ce quel que soit leur état. Il vaut donc mieux dans ce cas, une fois la base arrêtée, supprimer (ou mieux : renommer, ou archiver) l'ensemble des fichiers de la base de données avant de procéder à la restauration.

NB 2 : l'arborescence des fichiers doit exister préalablement à la restauration. RMAN ne recréera pas les répertoires manquants.

```
RMAN> startup force mount;
RMAN> restore database;
RMAN> recover database;
RMAN> alter database open;
```

10.1.4.4.3. Récupération incomplète de la base de données

- Pour effectuer une récupération incomplète de la base de données (c'est-à-dire remonter à un point dans le temps par ex.), il faut restaurer l'ensemble de la base de données puis effectuer la récupération incomplète.

La base doit donc être arrêtée (SHUTDOWN IMMEDIATE ou SHUTDOWN ABORT si l'arrêt normal pose problème), puis il faut supprimer (ou mieux : renommer, ou archiver) l'ensemble des fichiers de la base de données.

```
RMAN> startup force mount;
RMAN> run {
RMAN> set UNTIL TIME "to_date('11/05/2024 12:02', 'DD/MM/YYYY
                HH24:MI')";
RMAN> restore database;
RMAN> recover database;
RMAN> alter database open resetlogs;
RMAN> }
```

10.1.4.4.4. Récupération d'un fichier de base de données

- Il est possible de ne récupérer qu'un seul fichier de base de données.

Dans ce cas, au démarrage ou à l'arrêt de la base de données, un message d'alerte indique le fichier qui pose problème.

Dans ce cas, il est possible de ne restaurer que le fichier impacté :

```
RMAN> startup force mount;
RMAN> restore datafile 11;
RMAN> recover datafile 11;
RMAN> alter database open;
```

Rq : dans notre exemple, le fichier est indiqué par son identifiant (11) - Cet identifiant apparaît dans le message d'erreur.

10.1.4.4.5. Récupération à chaud d'un fichier de base de données ou d'un tablespace

- Pour limiter au maximum la rupture de service, il est possible de restaurer un fichier de données sans arrêter la base de données.

Si la base de données remonte le message suivant :

ORA-01122: échec de contrôle de vérification pour le fichier BdD 11
ORA-01208: ancienne version du fichier de données - pas accès a version en cours

Il est possible de se connecter à la base via RMAN, sans stopper la base de données :

```
RMAN> sql 'alter database datafile 11 offline';
RMAN> recover datafile 11;
RMAN> sql 'alter database datafile 11 online';
```

- Restauration d'un fichier de données : supprimer le fichier de données sur le disque puis taper la commande :

  "restore datafile nn" avant le recover.

- Pour restaurer un tablespace, les commandes sont sensiblement les mêmes :

```
RMAN> sql 'alter tablespace TBS_NAME offline';
RMAN> recover tablespace TBS_NAME;
RMAN> sql 'alter tablespace TBS_NAME online';
```

10.1.4.4.6. Récupération du fichier d'initialisation

- Il est toujours plus prudent de garder une copie du fichier d'initialisation afin de la restaurer en cas de perte.
- Dans le cas d'un spfile, on peut garder une copie d'un fichier init sur lequel on va démarrer la base ou RMAN.

10.1.4.4.7. Récupération d'un fichier de contrôle

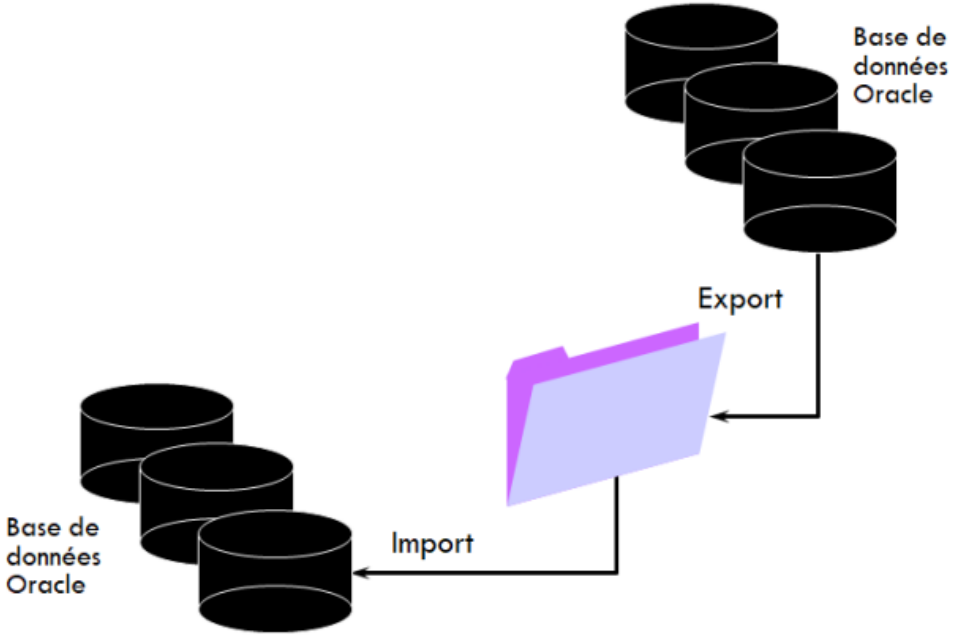
- Dans le cas de la perte d'un seul fichier de contrôle (ou en tout cas dans le cas où il en existe encore au moins une copie), le plus simple est d'arrêter la base, et de recopier l'une des copies du fichier de contrôle à l'emplacement attendu du fichier de contrôle manquant.
- Si tous les fichiers de contrôle ont été perdus : s'assurer que l'instance est bien arrêtée (SHUTDOWN ABORT si ce n'est pas le cas, car un IMMEDIATE ne marchera pas).

```
RMAN> startup force mount;
RMAN> restore controlfile from autobackup;
RMAN> alter database open Resetlogs;
```

Rq : cette manipulation suppose que les fichiers de contrôle étaient en mode AUTOBACKUP dans RMAN.

11. Transferts de données

- L'utilitaire export/import permet de sauvegarder le contenu logique d'une base de données dans un fichier de transfert Oracle au format binaire, ou fichier dump. Ce fichier pourra donc être relu pour recréer des objets qu'il contient.



11.1.Les différents modes d'export et d'import

a.Niveau base de données complète

- C'est le mode le plus complet, lors de ce type d'exportation tous les objets de la base sont exportés à l'exception de certains utilisateurs : SYS, ORDSYS, CTXSYS, MDSYS et ORDPLUGINS.
- Par contre les informations relatives aux structures de base de données, telles que les définitions de tablespaces et de segments de rollback, sont incluses.

Exemple :

- Export FULL des objets

```
C:\> exp userid=system/manager file=c:\backup\export_full.dump
     log=c:\backup\export_full.log full=y rows=n
```

Ici, on se connecte à la base en tant que SYSTEM (**userid=system/manager**) et on exporte toute la base (**full=y**) sans les données (**rows=n**). On sauvegarde la sortie dans le fichier de log (**log=c:\backup\export_full.log**).

- Import de tous les schémas inclus dans le DUMP

```
C:\> imp userid=system/manager file=c:\backup\export_full.dump
     log=c:\backup\export_full.log
```

b.Niveau utilisateur

- Tous les objets appartenant à un utilisateur sont exportés : tables, fonctions, synonymes, déclencheurs, liens de bases de données…
- Le paramètre OWNER permet de désigner les utilisateurs que l'on désire exporter et le paramètre FROMUSER désigne quel est l'utilisateur à importer du fichier Dump (le paramètre TOUSER nous indique quant à lui le schéma destinataire).

Exemples :

- Export du schéma SCOTT

C:\> exp userid=system/manager file=c:\backup\export_scott.dump
log=c:\backup\export_scott.log owner=hr

- Import du schéma HR

C:\> imp userid=scott/tiger file=c:\backup\export_scott.dump
log=c:\backup\export_scott.log owner=hr

- Import du schéma HR dans le schéma USER1

C:\> imp userid=system/manager file=c:\backup\export_hr.dump
log=c:\backup\export_hr.log fromuser=hr touser=user1

L'utilisateur de connexion (indiqué dans userid) doit naturellement être autorisé à créer les objets dans le schéma USER1.

c. Niveau table

- Lors de l'exportation de tables individuelles tous leurs objets associés (index, contraintes, déclencheurs, privilèges …) sont écrits dans le fichier DUMP.
- Lors de l'importation tout comme l'exportation les tables doivent être nommées grâce au paramètre TABLES.

Exemples :

- Export de la table DEPARTMENTS de l'utilisateur HR

```
C:\> exp userid=system/manager file=c:\backup\export_departments.dump
     log=c:\backup\export_departments.log tables=hr.departments
```

- Import de la table DEPARTMENTS de l'utilisateur HR dans USER1

```
C:\> exp userid=system/manager  file=c:\backup\export_departments.dump
    log=c:\backup\export_departments.log fromuser=hr touser=user1
    tables=departments
```

d.Niveau tablespace

- Lors de l'exportation, des métas donnés concernant les tablespaces spécifiés et les objets qu'ils contiennent sont écrites dans un fichier DUMP.
- les bases source et cible doivent avoir la même taille de bloc et même CHARACTER SET
- Avant de déplacer un tablespace, il doit être mis READONLY puis le schéma des données contenus dans le tablespace doit être exporté.
- Avantages des tablespaces transportables : Copie rapide des informations entre les bases de production, la base du DatawareHouse et les bases DataMart

- Etapes de déplacement d'un Tablespace :

  1. Vérification que le tablespace est transportable.

  SQL> EXECUTE sys.dbms_tts.transport_set_check('USERS', TRUE);

  SQL> SELECT * FROM sys.transport_set_violations;

  2. Création d'un pool de fichiers transportables : Un pool contient les fichiers du ou des tablespaces et le ou les fichiers contenant les métadonnées.

  Sql>ALTER TABLESPACE app_data READ ONLY;

  Exporter le schéma des données du ou des tablespaces

  C:> EXP TRANSPORT_TABLESPACE=y TABLESPACES=(app_data) TRIGGERS= y/n CONSTRAINTS= y/n GRANTS= y/n FILE=test.dmp

3. Transport du pool de fichiers : Copie des fichiers de données et d'exports du schéma vers la base cible (via les commandes de l'OS)

4. Attacher le tablespace à la base cible : faire un import pour attacher les fichiers de données à la base cible. Emplacement identique à la base source.

```
C:>IMP TRANSPORT_TABLESPACE=y
    DATAFILES=('f:\oracle\oradata\dbtests\test2\test_1.dbf')
    TABLESPACES=(app_data) TTS_OWNERS=(user1)
    FROMUSER=(user1) TOUSER=(hr) FILE=test.dmp
```

e. Exporter/Importer avec un fichier de paramètres

- les utilitaires d'export et import permettent d'indiquer un fichier de paramètres via l'option **parfile**.

  Avec le fichier **c:\backup\parfile.prm** suivant :

```
userid=system/manager
file=c:\exp_hr.dmp
log=c:\exp_hr_log.txt
owner=hr
rows=n
```

```
C:>exp parfile=c:\backup\parfile.prm
```

```
C:>imp parfile=c:\backup\parfile.prm
```

f. Exporter selon une condition

- Il est possible de n'exporter qu'une partie de la table et non pas la totalité. Il suffit de le préciser par l'option QUERY.

Exemple : exporter les personnes dont le salaire est supérieur à 500, dropper la table et enfin la réimporter.

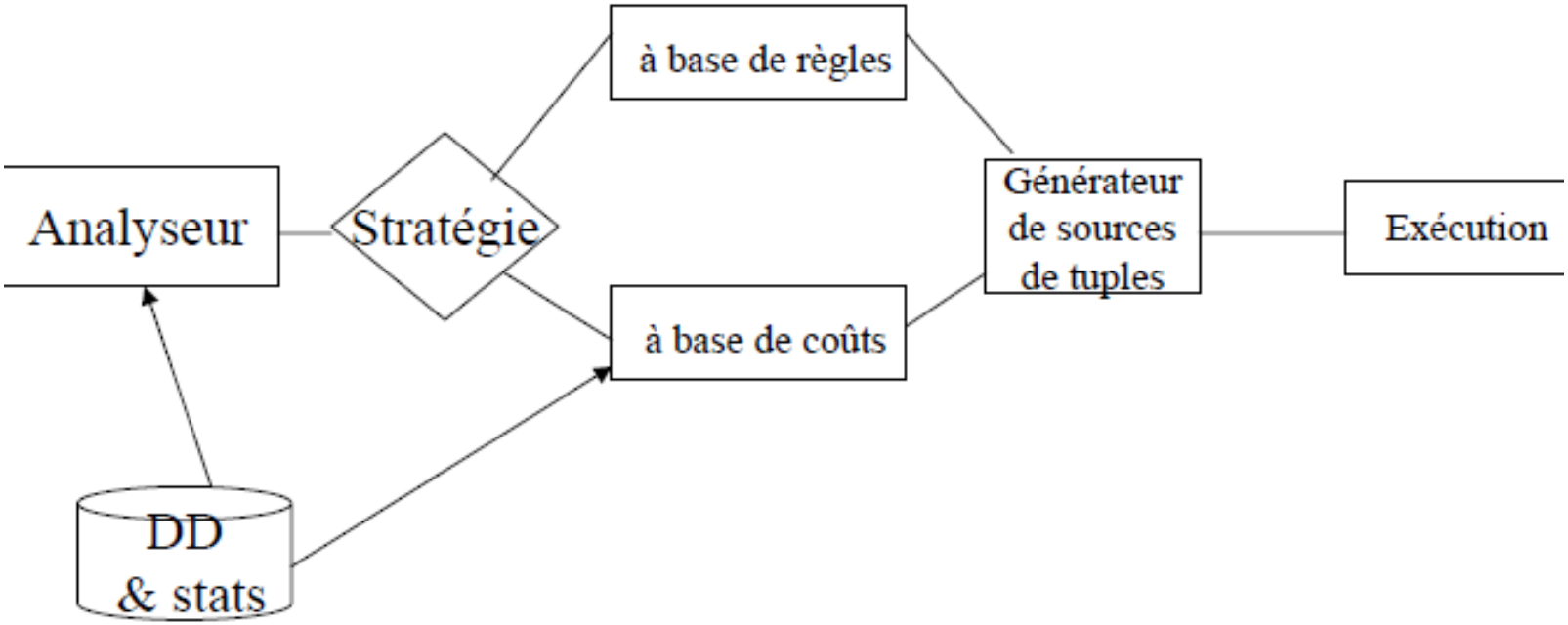
C:>exp system/manager file=/ora/admin/dba/log/exp_query.dmp tables=hr.employees query="'where sal > 5000'"

C:>imp system/manager file=/ora/admin/dba/log/exp_query.dmp fromuser=hr touser=user1 tables=employees log=log.imp

SQL> select * from employees;

12.Optimisation SQL

- Pour optimiser la vitesse de requête sur n'importe quel moteur SQL, il est essentiel de comprendre l'ordre d'exécution SQL.
- Déterminer la stratégie optimale d'exécution des opérations élémentaires :



- Rule-Based Optimizer (RBO) - Il s'agissait de la méthode d'optimisation d'origine et, comme son nom l'indique, il s'agissait essentiellement d'une liste de règles qu'Oracle devait suivre pour générer un plan d'exécution.
  - Règle n° 1 : Lecture des lignes isolées à l'aide du ROWID
  - …….
  - Règle n° 8 : Accès par l'intermédiaire d'un index composite avec toutes les clés contenues dans la clause where.
  - Règle n° 9 : Accès par l'intermédiaire d'un index sur une colonne.
  - …….
  - Règle n° 15 : Balayage complet de la table.
- Optimiseur basé sur les coûts (CBO) - Évaluer le coût d 'exécution de la requête en utilisant des statistiques sur :
  - nb exact ou estimé de tuples et de blocs par table
  - nb de niveaux d'index et nb de valeurs distinctes par attribut : sélectivité d 'un attribut = nb de val. différentes / nb tuples (1 pour les clés)
  - nb estimé de L/E logiques et physiques, ...

12.1.Mise en place d'une stratégie d'optimisation

L'optimisation des requêtes SQL dans Oracle est un processus visant à améliorer les performances des requêtes, en réduisant le temps d'exécution et en optimisant l'utilisation des ressources.

– **Conception appropriée de la base de données** : Une bonne conception de la base de données, avec des schémas normalisés et des index appropriés, peut améliorer considérablement les performances des requêtes.

– **Indexation appropriée** : Les index sont utilisés pour accélérer les recherches dans les tables. Création des index sur les colonnes fréquemment utilisées dans les clauses WHERE, JOIN et ORDER BY des requêtes.

– **Collecte et utilisation de statistiques** : permet à Oracle d'optimiser les plans d'exécution des requêtes en fonction des données réelles.

– **Utilisation de clauses JOIN appropriées** : Utilisation des clés étrangères pour faciliter les jointures entre les tables.

- **Utilisation de clauses WHERE efficaces** : Éviter d'utiliser des fonctions ou des opérations coûteuses dans les clauses WHERE, car cela peut empêcher l'utilisation d'index.
- **Optimisation des sous-requêtes** : Les sous-requêtes peuvent être coûteuses en termes de performances. Essayer de les optimiser en utilisant des jointures ou des opérations de regroupement plutôt que des sous-requêtes imbriquées.
- **Réécriture de requêtes** : Parfois, une requête peut être réécrite de manière à obtenir un plan d'exécution plus efficace. Expérimenter avec différentes formulations de requêtes pour trouver la plus performante.
- **Utilisation de l'outil d'optimisation d'Oracle** : Oracle fournit des outils tels que le planificateur de requêtes, le conseiller SQL et l'outil d'optimisation automatique (Automatic SQL Tuning) pour optimiser les requêtes. Utiliser ces outils pour analyser et améliorer les plans d'exécution des requêtes.

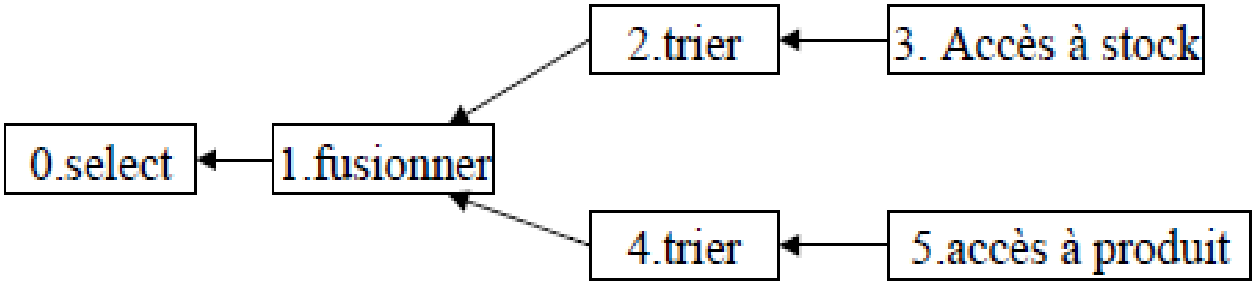
12.2. Outils d'analyse et de mesures

12.2.1. Explain Plan

- Le Plan d'exécution permet d'obtenir une vue détaillée du processus d'exécution des requêtes dans Oracle.
  - Utilise une table d'audit (PLAN_TABLE) pour recevoir les résultats de l'analyse - créée par le script utlxplan.sql.
- On peut n´néanmoins créer une table de même schéma que plan table et y mettre le plan engendré.
  - Dans ce dernier cas, on utilisera la clause INTO NomTable de EXPLAIN PLAN.

- Exemple d'un plan d'exécution :

Select p.libelle, s.qte from produit p, stock s where p.prodnum=s.prodnum;



Explain plan set statement_id=' XXX '
for select p.libelle ….

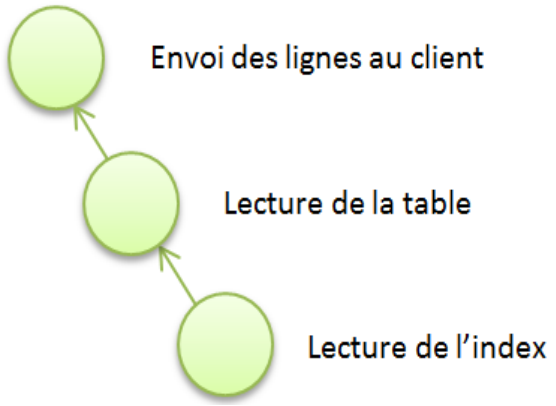
Puis select id, parent_id, operation, object_base,options from plan_table;

Id	parent_id	operation	object_base	options
0		select statement		
1	0	merge join		
2	1	sort		join
3	2	table access	stock	full
4	1	sort		join
5	4	table access	produit	full

- Lorsqu'on demande le contenu d'une table, Oracle va devoir lire les blocs de cette table. Mais dans certains cas, un index peut accélérer la lecture ; s'il l'utilise Oracle va donc lire l'index puis les blocs de la table référencés par l'index.

  - Le plan d'exécution comporte là 2 étapes :
    1) Lecture d'un index
    2) Lecture des blocs référencés par l'index

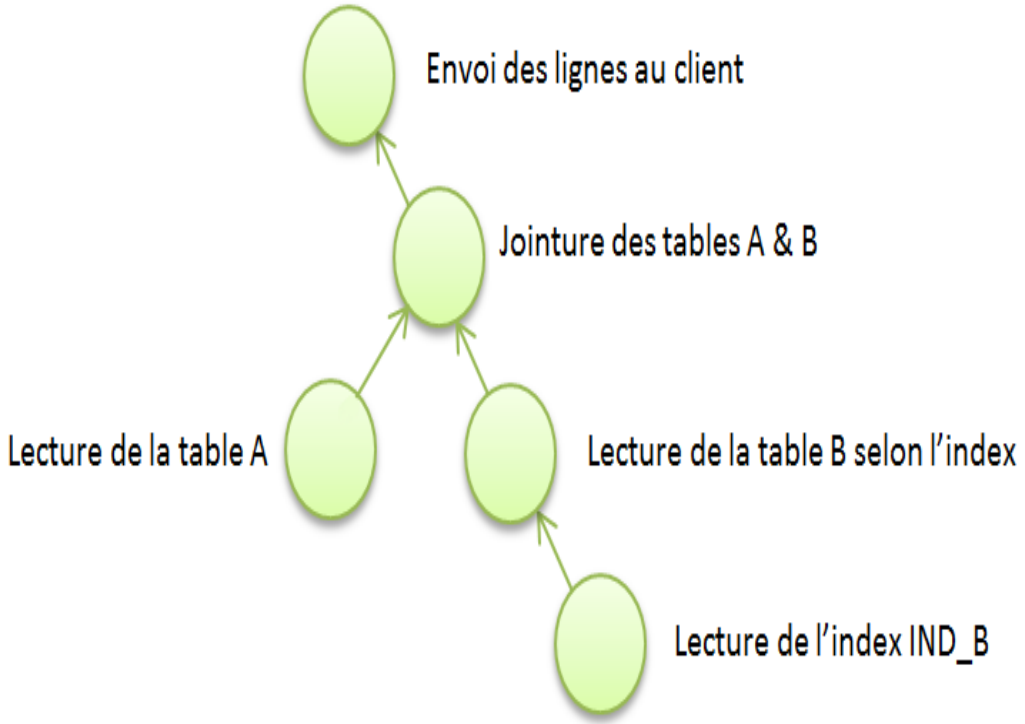
    Ce plan peut s'écrire sous forme d'arbre :



Voici un plan d'exécution tel qu'Oracle l'écrit en mode texte :

```
-------------------------------------------------------------------
| Id| Operation                    | Name
-------------------------------------------------------------------
| 0 | SELECT STATEMENT             |
| 1 | TABLE ACCESS BY INDEX ROWID  | CONSULTANT   |
| 2 |  INDEX RANGE SCAN            | CONSULTANT_PK
```

- Les plans d'exécution se complexifient lorsqu'on réalise des jointures.
  - Dans ce cas, Oracle ajoute une étape pour faire correspondre le contenu des blocs d'une table avec une autre.



– Mise en œuvre EXPLAIN PLAN :

Visualiser un plan d'exécution avec Oracle nécessite deux étapes :

1. Sauvegarder le plan d'exécution dans PLAN_TABLE avec explain plan for

2. Formater et afficher le plan d'exécution.

```
EXPLAIN PLAN [SET STATEMENT_ID = 'Nom de requête']
[INTO NomTable] FOR instruction SQL;
```

Instruction SQL:
SELECT, INSERT, UPDATE, DELETE, CREATE TABLE,
CREATE INDEX, ALTER INDEX ...

– Les plans étant ajoutés à la table plane_table, il faudra veiller :

- Soit à vider la table à chaque génération de plan,
- Soit à nommer les requêtes (dans explain plan : explain plan set statement_id ='Nom de la requête') INTO NomTable.

Exemple:

```
SQL>EXPLAIN PLAN FOR select * from dual;
SQL>select * from table(dbms_xplan.display);
Plan hash value: 272002086
--------------------------------------------------------------------------------
| Id  | Operation                 | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT          |      |     1 |     2 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL        | DUAL |     1 |     2 |     2   (0)| 00:00:01 |
```

Exploitation de la table PLAN

Opération	**Opérateur de base**
FILTER	*Sélection*
MERGE JOIN	*Jointure par tri - fusion*
NESTED LOOP	*Jointure par utilisation de l'indexation*
PROJECTION	*Projection*
TABLE ACCES FULL	*Accès séquentiel*
TABLE ACCESS BY ROWID	*Accès direct*
INDEX ….	*Utilisation de l'indexation*

Exemple :

```
SQL> EXPLAIN plan for
  UPDATE employees
  SET salary = salary * 1.10
  WHERE department_id = (select department_id from departments where
department_name='ADMINISTRATION');

SQL> select ID, OPERATION, OPTIONS, OBJECT_NAME, POSITION from
plan_table;
```

ID	OPERATION	OPTIONS	OBJECT_NAME	POSITION
0	UPDATE STATEMENT	(null)	(null)	4
1	UPDATE	(null)	EMPLOYEES	1
2	INDEX	RANGE SCAN	EMP_DEPARTMENT_IX	1
3	TABLE ACCESS	FULL	DEPARTMENTS	1

Avec le plan d'exécution, d'autres informations sont données comme le coût d'une étape ou bien le temps passé sur celle-ci.

12.2.2. Génération des statistiques

- Résultats stockés dans le dictionnaire de données
  - xxx_TABLES, xxx_TAB_COLUMNS
  - xxx_INDEXES, xxx_CLUSTERS

```
SQL>ANALYSE TABLE hr.employees option STATISTICS;
```

Options:

- COMPUTE : calcul à partir de toutes les données
- ESTIMATE : Calcul avec échantillonnage
- DELETE : suppression des statistiques

```
SQL>SELECT global_stats FROM dba_tables where
    table_name='HR.EMPLOYEES';
```

- Les différentes méthodes de calcul des statistiques d'objets
    - Calcul des stats pour la table EMPLOYEES et tous ses index

```
SQL>ANALYSE TABLE hr.employees compute STATISTICS;
```

    - Calcul des stats pour toutes les colonnes de tous les index de la table EMPLOYEES

```
SQL>ANALYSE TABLE hr.employees compute STATISTICS for all indexed columns;
```

    - Calcul des stats pour la table EMPLOYEES et tous ses index sur un échantillon de 20% des lignes)

```
Analyze table EMPLOYEES estimate statistics sample size 20 percent
```

- La procédure analyze_schema du package DBMS_UTILITY provoque le calcul des stats pour l'ensemble du schéma.

```
Execute dbms_utility.analyze_schema( 'HR', 'estimate', estimate_percent=>20) ;
```

- Package DBMS_STATS : Les statistiques de table peuvent être collectées pour la base de données, le schéma, la table ou la partition.

```
EXEC DBMS_STATS.gather_database_stats;
EXEC DBMS_STATS.gather_database_stats(estimate_percent => 15);
EXEC DBMS_STATS.gather_database_stats(estimate_percent => 15, cascade => TRUE);

EXEC DBMS_STATS.gather_schema_stats('HR');
EXEC DBMS_STATS.gather_schema_stats('HR', estimate_percent => 15);
EXEC DBMS_STATS.gather_schema_stats('HR', estimate_percent => 15, cascade =>
TRUE);

EXEC DBMS_STATS.gather_table_stats('HR', 'EMPLOYEES');
EXEC DBMS_STATS.gather_table_stats('HR', 'EMPLOYEES', estimate_percent => 15);
EXEC DBMS_STATS.gather_table_stats('HR', 'EMPLOYEES', estimate_percent => 15,
cascade => TRUE);

EXEC DBMS_STATS.gather_dictionary_stats;
```

12.3.Éléments d'optimisation : Vue, Index, Cluster

12.3.1. VUE

- Table VIRTUELLE
- Requête stockée dans la base de données
- Construite sur table(s) et / ou vue(s)
- Utilisée comme une table

- Utilités ?

- Gérer la confidentialité des données
- Préparer des requêtes complexes
- Présenter plusieurs visions différentes des mêmes informations
- Maîtriser les mises à jour

12.3.2. INDEX

- Un index est un objet supplémentaire créé sur une ou plusieurs colonnes d'une table pour faciliter un accès rapide aux données.

- Il contient un certain volume d'informations (overhead) dont le coût de traitement peut dépasser celui d'un balayage complet de la table, selon le nombre de blocs qu'il faut lire dans la table pour renvoyer la ligne pointée par le ROWID de l'index.

- La structure de données utilisée par Oracle pour stocker un index est un B*-tree, y compris pour les index de type bitmap.

- Construire des index optimaux. Quelques options d'indexation :
  - Non-unique index (index avec doublons)
  - Unique index (index sans doublons)
  - Bitmap index
  - Local prefixed partitioned index (index de partitionement local préfixé)
  - Local nonprefixed partitioned index (index de partitionement local non préfixé)
  - Global prefixed partitioned index (index de global local préfixé)
  - Hash partitioned index
  - Composite partitioned index
  - Reverse-key index
  - Function-based index
  - Descending index
  - Index-organized-table

Création d’index

```
CREATE INDEX index_name
ON table_name (column1, column2, ...)
```

- index_name : spécifie le nom que vous souhaitez donner à l'index.
- table_name : indique le nom de la table sur laquelle l'index sera créé.
- column1, column2, ... : spécifie les noms des colonnes de la table sur lesquelles vous souhaitez créer l'index. Vous pouvez spécifier une ou plusieurs colonnes séparées par des virgules.