2008

UFR Ingénieurs 2000

Vivien Boistuaud
Julien Herr

ADMINISTRATION DE BASES DE DONNEES ORACLE

Ce document a été réalisé par V. Boistuaud et J. Herr dans le cadre des travaux pratiques d'administration des bases de données Oracle dispensés par J.P. Squelbut, administrateur senior de bases de données Oracle.

Sommaire

Introduction

Suite aux cours d'administration de bases de données Oracle assurés par J.P. Squelbut, nous avons réalisé une séance de travaux pratiques de 8h. Celle-ci a été divisé en deux parties indépendantes l'une de l'autre.

La première consistait à nous familiariser avec l'environnement Oracle ainsi que ses outils Web (console d'administration web) et de commandes (`sqlplus`). L'objectif final était de mettre en place un script permettant l'introduction de plusieurs DDL[1] tout en les isolants les uns des autres. Un administrateur de bases de données peut être couramment confronté à cette situation dans la mesure où elle est généralement consécutive à la création d'un nouveau projet utilisant une base de données, par exemple la conception d'une nouvelle solution logicielle.

La seconde partie consistait à nous mettre dans la peau d'un administrateur chargé de résoudre les problèmes rencontrés sur une base de données, provoquant des ralentissements d'un applicatif.

[1] Database Description Language (tables, indexes, ...)

1. Mode opératoire

Dans le cadre de cette séance de Travaux Pratiques, nous avons utilisé le matériel suivant :

- Une machine Linux par utilisateur, équipé du client SSH et d'un navigateur Web,
- Un serveur Linux hébergeant les applications Oracle, accessible par SSH

L'architecture mise en œuvre n'est pas idéale. En effet, le serveur Linux héberge à la fois le SGBD Oracle, les outils d'administration, et la console OEM Oracle. Etant donné les moyens mis à notre disposition, une seule base de données était partagée entre tous les utilisateurs (9 personnes par séance) et l'activité de la console OEM était susceptible de perturber le bon fonctionnement du SGBD, car elle consomme des ressources systèmes sur la machine serveur.

Cependant, ceci n'est pas nécessairement gênant, bien que ce type de situation soit très rare en entreprise.

Pour se connecter au serveur SSH depuis les postes clients, il faut demander une connexion vers l'adresse IP associée au nom de domaine local `bdd.ig2k.umlv`. Ceci se fait simplement en utilisant la commande :

```
# ssh bdd.ig2k.umlv
```

Les logins et mot de passe que nous avons utilisé pour la connexion étaient les comptes `dba02` et `dba09`.

La version d'Oracle utilisée pour ce TP était Oracle 10i, qui est l'avant dernière version en date lors de l'écriture de ce document.

2. Découverte de l'administration Oracle

Les objectifs de ce TP étaient de découvrir le scripting d'administration avec Oracle, de découvrir la console d'administration OEM Oracle et de réfléchir sur l'organisation des « `user` », « `schema` », « `tablespace` » et « `privilege` » fournis par la solution de bases de données Oracle.

Pour cela, nous nous sommes documentés afin de mettre en place en script permettant l'introduction de plusieurs DDL tout en les isolants les uns des autres.

En entreprise, il est fréquent de ne créer qu'une seule base de données Oracle, et de définir autant de tablespaces que d'applications souhaitant utiliser le SGBD. Notre mission revient donc à créer un script permettant de mettre en place automatiquement un nouveau tablespace et tous les éléments associés (utilisateur, droits…) de sorte que les développeurs puissent utiliser indépendamment le SGBD pour chaque projet.

1. Sujet de l'exercice

```
From squelbut@bdd.univ-mlv.fr Wed Jan 23 08:40:22 2008
Envelope-to: dba02@bdd.univ-mlv.fr
Delivery-date: Wed, 23 Jan 2008 08:40:22 +0100
To: dba01@bdd.univ-mlv.fr,dba02@bdd.univ-mlv.fr,
        dba03@bdd.univ-mlv.fr,dba04@bdd.univ-mlv.fr,
        dba05@bdd.univ-mlv.fr,dba06@bdd.univ-mlv.fr,
        dba07@bdd.univ-mlv.fr,dba08@bdd.univ-mlv.fr,
        dba09@bdd.univ-mlv.fr,dba10@bdd.univ-mlv.fr,
        dba11@bdd.univ-mlv.fr,dba12@bdd.univ-mlv.fr
Subject: consigne TP1
From: Jean-Philippe Squelbut <squelbut@bdd.univ-mlv.fr>
Date: Wed, 23 Jan 2008 08:40:22 +0100

1) L'ENVIRONNEMENT
Vous travaillez en tant que DBA, vous travaillez donc sur le serveur.
(En outre le poste sur lequel vous travaillez ne possede pas
d'installation client Oracle.) Les serveur a contacter est : bdd.ig2k.umlv.
Pour pouvoir contacter la base Oracle,
il faut positionner un certain nombre de variables
d'environnement correspondant a l'installation.
Soit dans votre profile, soit par un fichier de commande. Les voici :
-----------------------------------------------------------------
export ORACLE_HOME=/app/oracle/product/10.2.0/db_1
export
PATH=/app/oracle/product/10.2.0/db_1/bin:/app/oracle/product/10.2.0/db_1/
bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
export
LD_LIBRARY_PATH=/app/oracle/product/10.2.0/db_1/lib:/app/oracle/product/1
0.2.0/db_1/lib:
export ORACLE_SID=TOPO
export NLS_LANG=AMERICAN_FRANCE.WE8ISO8859P1
export EDITOR=vi
-----------------------------------------------------------------

2) La base TOPO
Elle est unique pour tous les groupes. Si vous l'arretez, les autres groupes
ne travaillent plus.

3) Le TP : scripter ce qu'il faut pour
 - introduire plusieurs DDL dans la base
 - isoles les uns des autres.

4) Les objectifs du TP sont :
 - la decouverte du scripting d'administration avec Oracle
 - la decouverte de la console OEM Oracle
 - la reflexion sur l'organisation user, schema, tablespace,les privileges
 - la remise d'un compte-rendu qui argumente vos choix d'organisation.

5) La methode
 - se documenter, scripter, tester
 - essayer par la console OEM
   et recuperer les scripts SQL pour les adapter
   (la console est disponible en http://bdd.ig2k.umlv:5502/em/
       user system password ig2k)

Bon courage.
```

2. Mise en place de l'environnement

Une seule base de données (baptisée TOPO) était disponible pour l'ensemble des DBA que nous représentions. Pour pouvoir l'administrer par la console, il fallait se connecter à la machine `bdd.ig2k.umlv` qui héberge le SGBD et la base de données TOPO.

```
# cat > ~/.bashrc
export ORACLE_HOME=/app/oracle/product/10.2.0/db_1
export
PATH=/app/oracle/product/10.2.0/db_1/bin:/app/oracle/product/10.2.0/db_1/bin:/
usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
export
LD_LIBRARY_PATH=/app/oracle/product/10.2.0/db_1/lib:/app/oracle/product/10.2.0
/db_1/lib:
export ORACLE_SID=TOPO
export NLS_LANG=AMERICAN_FRANCE.WE8ISO8859P1
export EDITOR=vi
^D
# source ~/.bashrc
```

Le script ci-avant est à placer sur le serveur de base de données. L'intérêt de modifier le fichier .bashrc est d'avoir la configuration opérationnelle automatiquement à chaque connexion.

Ces différentes variables d'environnement permettent de configurer automatiquement les outils Oracle comme sqlplus, et de les rendre accessibles dans le path pour une exécution simplifiée.

3. Découverte des l'interfaces d'administration d'Oracle

Nous avons également pu découvrir l'interface d'administration web d'Oracle durant ce TP. Celle-ci était disponible à l'adresse web suivante : http://bdd.ig2k.umlv:5502/em/. Le nom d'utilisateur de connexion était « `system` » et le mot de passe était « `ig2k` ».

Il est également possible d'accéder à un terminal exécutant les requêtes SQL désirées. Il s'agit de la console SQLPlus accessible sur le serveur par la commande `sqlplus / as sysdba`. Le / as sysdba précise que nous souhaitons bénéficier des droits DBA associés à notre compte système UNIX. Par conséquent, aucune authentification de notre part ne sera nécessaire.

Il est intéressant de noter que les commentaires SQL commencent par « `--` » et que le mot clé « `prompt` » permet d'afficher le texte qui suit dans la console SQLPlus. Il s'agit d'un équivalent du « `echo` » utilisé en shell scripting.

Figure 1 - Page d'administration de la base de données

Cette interface est très complète et permet à la fois de détecter les problèmes du serveur et/ou du SGBD, mais surtout d'administrer la base de données sans saisir de commandes : ajout/suppression de comptes, de tablespaces, de tables, de vues…

Cependant, lors d'une création massive ou fréquente de nouveaux éléments en base de données, ce type d'interface peut s'avérer relativement fastidieux. En effet, pour créer un tablespace, il faut par exemple consulter 5 écrans différents avant que cette opération ne soit réalisée.

Dans un premier temps, nous avons cependant utilisé cette console pour :

- Découvrir le DDL associé aux opérations courantes (création de comptes, de tablespaces…)
- Se familiariser avec les puissants outils de diagnostics fournis par Oracle, et qui en font un atout face à ses principaux concurrents (DB2, Microsoft SQL Server, PostgreSQL…).

4. Création du tablespace

Via l'interface d'administration web et en s'appuyant sur la documentation Oracle, nous avons cherché la requête SQL permettant de créer un espace de table, ou tablespace en Anglais. Chaque base de données peut posséder plusieurs tablespaces, qui correspondent à un ensemble distinct de tables, index, … Ainsi, avec une seule base de données Oracle on peut héberger plusieurs espaces de travail logiciels distincts. Au niveau du système d'exploitation, la création d'un tablespace ne modifie pas le nombre de processus alloués, contrairement à la création d'une nouvelle base de données.

Lors de nos recherches, nous nous sommes rendu compte que pour créer un utilisateur, il fallait tout d'abord créer un tablespace. C'est ce que nous avons fait avec la requête suivante :

```
#
# Variables personnalisant le script
#
username=test_user
tblsuffix=tpl
password=defaultpass
size=10M

-- Creating the tablespace

prompt Tablespace Creation - CREATE SMALLFILE TABLESPACE
"${username}_${tblsuffix}" DATAFILE '/data/TOPO/${username}.db' SIZE ${size}
AUTOEXTEND ON NEXT 20M MAXSIZE UNLIMITED LOGGING EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

CREATE SMALLFILE TABLESPACE "${username}_${tblsuffix}" DATAFILE
'/data/TOPO/${username}.db' SIZE ${size} AUTOEXTEND ON NEXT 20M MAXSIZE
UNLIMITED LOGGING EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO;
```

L'utilisation de variables permet d'avoir un script générique réutilisable quel que soit le nom de l'utilisateur que nous voulons ajouter ou son mot de passe.

De plus, afin de mettre en place une stratégie de nommage des tables, nous avons choisis d'appeler les tablespaces par le nom du user préfixé par « _tbl ». Cette normalisation du nommage des tablespaces n'est pas anodine. En effet, les tablespaces sont représentés physiquement par des fichiers ; il est donc beaucoup plus aisés de retrouver ces fichiers s'ils ont un nom explicite, et de retrouver l'utilisateur principal associé si son nom est contenu dans le nom du tablespace.

Enfin, nous avons définis les valeurs des attributs suivants :

- DATAFILE : chemin physique du fichier sur le disque ;
- SIZE : la taille du tablespace lors de sa création ;
- AUTOEXTEND ON NEXT 20M : pour que le tablespace s'agrandisse automatiquement quand il est plein, et qu'il augmente par tranches de 20M pour éviter les réallocations fréquentes. Cette valeur pourrait être placée en variable, en fonction des quantités de données insérées dans la base par l'applicatif et de la fréquence d'ajout ; ceci n'est évidemment possible que si les développeurs ont fait un plan de stockage ;
- MAXSIZE UNLIMITED : limite la taille maximum du tablespace. Ici, un tablespace pourra avoir une taille illimitée, ou plutôt, limitée par l'espace de stockage disponible,

5. Création d'un utilisateur

Via l'interface d'administration web et en s'appuyant sur la documentation Oracle, nous avons cherché la requête SQL permettant de créer un utilisateur de la base.

Nous utilisons la requête suivante :

```
-- Creating main user

prompt User Creation - CREATE USER "${username}" PROFILE "DEFAULT" IDENTIFIED
BY "${password}" PASSWORD EXPIRE DEFAULT TABLESPACE "${username}_${tblsuffix}"
TEMPORARY TABLESPACE "TEMP" ACCOUNT UNLOCK;

CREATE USER "${username}" PROFILE "DEFAULT" IDENTIFIED BY "${password}"
PASSWORD EXPIRE DEFAULT TABLESPACE "${username}_${tblsuffix}" TEMPORARY
TABLESPACE "TEMP" ACCOUNT UNLOCK;
```

Nous avons définis les valeurs suivantes :

- PROFILE « DEFAULT » : détermine que le profil pris sera celui par défaut ;
- PASSWORD EXPIRE: signifie que le mot de passe s'expirera suivant les conditions définies par défaut ;
- DEFAULT TABLESPACE : Spécifie le nom du tablespace par défaut pour l'utilisateur. Si lors de sa connexion celui-ci ne précise pas le tablespace à utiliser, il sera automatiquement considéré comme placé sur l'espace de table indiqué après cette directive ;

- TEMPORARY TABLESPACE « TEMP » : spécifie que le tablespace pour le stockage des éléments temporaires sera celui baptisé « TEMP ». En l'occurrence, TEMP est un espace temporaire commun à tous les utilisateurs de la base ;

- ACCOUNT UNLOCK : permet d'activer le compte, le rendant utilisable directement. Si cette directive est absente, le compte sera créé mais désactivé par défaut : ceci peut permettre de faire des modifications de droit avant l'ouverture du compte, évitant ainsi l'exposition à des failles de sécurité potentielles.

6. Gestion des droits de l'utilisateur

Par défaut, un utilisateur n'a pas de droits. Il ne peut donc pas créer de base de données, ni de tablespace, et il ne peut ni consulter ni modifier d'informations dans la base TOPO ni aucun de ses tablespaces. Nous sommes dans le cadre d'une configuration d'accès par autorisation : tout est interdit sauf ce qui est explicitement autorisé.

Ceci étant justement ce que nous cherchions à avoir, nous avons donc donné un certain nombre de droit à l'utilisateur en complétant le script de la section précédente avec les lignes suivantes :

```
-- Change password

prompt alter user dbsnmp identified by "changepassplz";

-- Grant Quotas

prompt ALTER USER "${username}" QUOTA 100M ON "${username}_${tblsuffix}";

ALTER USER "${username}" QUOTA 100M ON "${username}_${tblsuffix}";

-- Grant connection rights

prompt GRANT "CONNECT" TO "${username}";

GRANT "CONNECT" TO "${username}";

-- Grant creation rights

prompt GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, CREATE SEQUENCE TO
"${username}";

GRANT CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, CREATE SEQUENCE TO
"${username}";
```

Les différentes commandes utilisées dans cette partie de script ont les significations suivantes :

- ALTER USER : Permet de définir qu'on va modifier les propriétés d'un utilisateur. Plus généralement, ALTER est une commande SQL qui permet de modifier un élément du SGBD comme une TABLE, un INDEX ou en l'occurrence un utilisateur ;
- IDENTIFIED BY : Permet de changer le mot de passe de l'utilisateur ;
- QUOTA 100M ON : Ici, on est en train de fixer le quota maximum de données que l'utilisateur va pouvoir stocker dans un tablespace donné ; dans notre cas, on autorise un stockage maximum de 100M sur le tablespace créé spécialement pour l'utilisateur. Cette valeur (de 100 Mo) pourrait être placée en variable, pour les besoins spécifiques des utilisateurs. 100Mo suffisent souvent si la base de données est utilisée dans le cadre d'un développement ;
- GRANT CONNECT TO : permet de donner le droit de se connecter à un utilisateur particulier sur la base. Sans ce droit, même si le compte utilisateur est actif, la connexion de ce dernier sera impossible ;
- GRANT CREATE TABLE, CREATE VIEW…: Donne à l'utilisateur le droit de créer des tables, vues, procédures stockées et séquences à condition qu'elles soient stockées dans un tablespace dans lequel ils sont accrédités.

L'utilisateur peut désormais créer des tables, des vues, des procédures stockées et des séquences dans le tablespace qui lui est défini.

Il a bien entendu le droit de se connecter à la base par l'intermédiaire de son login et mot de passe spécifié précédemment.

3. Jeu de rôle : Dans la peau d'un DBA

La deuxième partie de cette séance qui nous a été proposée, consistait à nous mettre la peau d'un DBA mandaté pour investiguer les dysfonctionnements d'une base de données Oracle à laquelle un logiciel est actuellement en train d'accéder de façon répétitive ; c'est-à-dire détecter et corriger les problèmes qui peuvent être rencontrés sur une base de données en production.

Ce cas arrive souvent en entreprise, et les problèmes peuvent aussi bien provenir du logiciel que du SGBD : dans ce cas, généralement, les deux services se rejettent la responsabilité du problème. Grâce aux outils fournis par Oracle, nous avons pu constater l'efficacité des systèmes de diagnostics mis en œuvre. Ceci représente un atout pour le SGBD Oracle, car le diagnostic est simplifié et le DBA apporte donc des réponses et des solutions efficaces plus rapidement.

Pour cet exercice, la base que nous avions à utiliser s'est volontairement retrouvée avec plusieurs problèmes que nous devions corriger. Cette section présente, dans un premier temps, le sujet du TP et, dans un second temps, les problèmes rencontrés et les outils utilisés pour les détecter.

1. Sujet de l'exercice

```
From squelbut@bdd.univ-mlv.fr Wed Jan 23 14:12:41 2008
Envelope-to: dba02@bdd.univ-mlv.fr
Delivery-date: Wed, 23 Jan 2008 14:12:41 +0100
To: dba01@bdd.univ-mlv.fr,dba02@bdd.univ-mlv.fr,
        dba03@bdd.univ-mlv.fr,dba04@bdd.univ-mlv.fr,
        dba05@bdd.univ-mlv.fr,dba06@bdd.univ-mlv.fr,
        dba07@bdd.univ-mlv.fr,dba08@bdd.univ-mlv.fr,
        dba09@bdd.univ-mlv.fr,dba10@bdd.univ-mlv.fr,
        dba11@bdd.univ-mlv.fr,dba12@bdd.univ-mlv.fr
Subject: sujet TP 1ere partie
From: Jean-Philippe Squelbut <squelbut@bdd.univ-mlv.fr>
Date: Wed, 23 Jan 2008 14:12:40 +0100

Une application tourne.
Les utilisateurs se plaignent.
Les temps sont tres longs.
Bien sur, selon la litanie habituelle, c'est la base Oracle qui est mise en
cause.
Vous devez intervenir.

Prevenants, ou avertis (ce n'est pas toujours le cas),
les utilisateurs vous indiquent les coordonnees de la base :
   la base est sur le serveur "bdd.ig2k.umlv", son SystemIDentifier est DBXO.
Les variables d'environnement necessaires pour s'y connecter.
export ORACLE_HOME=/app/oracle/product/10.2.0/db_1
export
PATH=/app/oracle/product/10.2.0/db_1/bin:/app/oracle/product/10.2.0/db_1/
bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
export
LD_LIBRARY_PATH=/app/oracle/product/10.2.0/db_1/lib:/app/oracle/product/1
0.2.0/db_1/lib:
export ORACLE_SID=DBXO
export NLS_LANG=AMERICAN_FRANCE.WE8ISO8859P1
export EDITOR=vi


1) Pour diagnostiquer le probleme, vous utiliserez :
   - les outils Unix a votre disposition
   - "perfstat" dont votre collegue vous dit qu'IL EST DEJA INSTALLE sur cette
base.
Pour ne pas passer pour un ignorant, vous ne lui demandez pas ce qu'est ce
"perfstat", ni comment l'utiliser.
La documentation est consultable par view
/app/oracle/product/10.2.0/db_1/rdbms/admin/spdoc.txt


Vous avez 2 heures devant vous.

2) Dans un rapport ecrit ulterieur, vous indiquerez aux utilisateurs :
   - si le probleme est reel ou pas,
   - vos observations,
   - vos propositions d'actions correctrices eventuelles.
```

2. Diagnostique du problème

La première étape consiste à trouver l'origine du problème. Pour cela, Oracle fournis des outils afin d'aider les DBA à analyser l'état des bases de données et diagnostiquer le problème. L'outil que nous avions utilisé était `perfstat`.

a. Top et uptime

Les premières investigations à réaliser pour s'assurer que le problème provienne de la base est d'investiguer au niveau du système d'exploitation lui-même. Pour cela, deux outils principaux sont à connaître : uptime, qui permet de savoir depuis combien de temps la machine est démarrée et quelle est la charge moyenne sur la dernière minute, les 5 dernières minutes et le dernier quart d'heure ; ainsi que la commande top, qui permet de contrôler la table des processus UNIX.

Après analyse, nous avons remarqué que la charge moyenne de la base sur le dernier quart était de 4.0, ce qui signifie qu'en moyenne, 4 processus sont en attente d'accès au processeur à chaque instant. Si la machine possède un seul processeur, alors cette valeur est assez élevée. Ici, la machine ne possède bien qu'un seul processeur.

De plus, dans la table des processus, deux processus prennent le plus de ressources : un processus de connexion au SGBD Oracle, et un processus java. Cependant, ceux-ci ne suffisent pas à eux seuls à saturer la machine. De plus, leur nature tend à nous indiquer qu'ils ont toutes les raisons de tourner sur la machine. En conséquence, le problème ne vient certainement pas du système d'exploitation, mais des processus qui consomment trop de ressources.

Comme les utilisateurs tendent à indiquer que ceci est inhabituel, et que le processus Oracle est très utilisé, il apparaît normal d'investiguer le SGBD plus en détail. Les craintes des utilisateurs semblent bien provenir d'Oracle.

b. Perfstat

Perfstat est un logiciel qui permet de fournir des rapports en fonction des données qui lui sont fournis. Les différentes données sont stockées dans les tables du schéma `perfstat`. Il permet d'étudier l'état de la base durant une période précise. De plus, il permet des analyses très fines puisqu'il va jusqu'au niveau des requêtes SQL, et de la façon dont Oracle les gère (hachage, durée, nombre d'exécution…).

Pour utiliser Perfstat, nous allons créer deux « snapshot » qui représentent l'état de la base de données au moment ou le « snapshot » a été effectué. Nous pouvons visualiser tous les « snapshot » de la base en utilisant la commande suivante via SQLPlus :

```
SQL> select SNAP_ID, SNAP_TIME from STATS$SNAPSHOT;

   SNAP_ID SNAP_TIM
---------- --------
        22 23/01/08
        23 23/01/08
         1 23/01/08
         2 23/01/08
        11 23/01/08
        12 23/01/08
        21 23/01/08

7 rows selected.
```

Durant ce TP, l'ensemble des étudiants travaillaient sur la même base de données. Ainsi, chacun pouvait effectuer des snapshots et ceux-ci étaient enregistrés dans la même table.

Nous avons donc effectué un snapshot à un instant t, puis nous avons attendu quelques minutes afin de prendre un deuxième snapshot. Afin de générer un rapport nous avons utilisé la commande suivante :

```
SQL> @?/rdbms/admin/spreport
```

Cette commande est en fait un script disponible à l'installation de la base Oracle et qui est chargé par l'intermédiaire du caractère « @ ». Il est possible de charger n'importe quel script de cette manière. Le caractère « ? » quant à lui signifie que le répertoire courant est le répertoire d'installation de la base Oracle.

Après cette commande, nous avons précisé le snapshot de début et de fin en précisant les identifiants de ceux-ci. Puis nous avons ensuite donné un nom au fichier de rapport qui sera généré par le script. Nous avons ensuite deux possibilités pour visualiser le rapport ainsi généré. La première consiste à utiliser un éditeur de texte tel que vi pour ouvrir le fichier. La deuxième s'effectue à partir de sqlplus via une commande :

```
SQL> !view <nomfichier>
```

La première partie du rapport comporte le Statspack qui permet d'avoir les informations de la base de données. Nous retrouvons par exemple son identifiant, la durée entre les deux snapshots, les buffers, la mémoire, la charge, …

```
STATSPACK report for

Database    DB Id    Instance     Inst Num Startup Time    Release     RAC
~~~~~~~~ ---------- ------------ -------- --------------- ----------- -
           969001921 DBXO                1 23-Jan-08 14:12 10.2.0.1.0  NO

Host  Name:   bdd              Num CPUs:    1        Phys Memory (MB):
0
~~~~

Snapshot       Snap Id     Snap Time      Sessions Curs/Sess Comment
~~~~~~~~    ---------- ------------------ -------- --------- -----------
Begin Snap:         11 23-Jan-08 15:54:44       48      10.2
  End Snap:         51 23-Jan-08 16:10:24       48      12.2
   Elapsed:               15.67 (mins)

Cache Sizes                      Begin        End
~~~~~~~~~~~                   ---------- ----------
               Buffer Cache:       420M       412M  Std Block Size:         8K
           Shared Pool Size:        64M        72M      Log Buffer:     2,859K

Load Profile                            Per Second       Per Transaction
~~~~~~~~~~~~                            ---------------  ---------------
                  Redo size:             11,031.14            157,110.12
              Logical reads:             37,119.32            528,669.14
              Block changes:                 22.14                315.29
             Physical reads:              1,873.75             26,686.74
            Physical writes:                  1.69                 24.09
                 User calls:                 19.53                278.11
                     Parses:                 24.46                348.39
                Hard parses:                  3.58                 51.02
                      Sorts:                 12.84                182.92
                     Logons:                  0.31                  4.36
                   Executes:                 45.97                654.71
               Transactions:                  0.07

  % Blocks changed per Read:    0.06    Recursive Call %:    95.81
 Rollback per transaction %:    0.00       Rows per Sort:    10.26

Instance Efficiency Percentages
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            Buffer Nowait %:   96.69       Redo NoWait %:  100.00
            Buffer  Hit   %:   94.95    In-memory Sort %:  100.00
            Library Hit   %:   89.51        Soft Parse %:   85.36
         Execute to Parse %:   46.79         Latch Hit %:   99.98
Parse CPU to Parse Elapsd %:   19.91     % Non-Parse CPU:   97.69

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   98.90   93.69
    % SQL with executions>1:   84.31   76.20
  % Memory for SQL w/exec>1:   96.23   93.98
```

La lecture de ces informations permet certaines déductions utiles à la localisation du problème. En effet, il y a beaucoup d'écriture dans les journaux et on constate près de 37 000 lectures par secondes (Logical reads).

Nous constatons également que dans 96% des cas, il n'y a pas de buffer, les informations sont déjà en mémoire.

Une autre partie du rapport permet de visualiser les 5 événements les plus consommateurs en temps. Dans le rapport que nous avons généré, nous avons obtenus les informations suivantes :

```
Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                   wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- -----
read by other session                      1,155,790      14,762     13   82.8
db file scattered read                       148,700       1,538     10    8.6
CPU time                                                      588           3.3
latch: cache buffers chains                   27,722         410     15    2.3
db file sequential read                       11,557         167     14     .9
```

Dans le cas d'une exécution normale, il est logique d'avoir le moins d'entrée/sortie possible. De plus, la consommation CPU doit en théorie est le paramètre le plus important lorsque la base est active. Dans ce cas précis, nous constatons que la consommation CPU est très faible par rapport aux autres paramètres. Beaucoup de temps est accordé à la lecture d'informations.

Les entrées/sorties sont beaucoup trop couteuses dans ce cas puisqu'elles représentent près de 90% du temps. Nous pouvons en déduire qu'il y a un problème au niveau de la SGA[2].

Par la suite, le rapport contient une analyse plus fine en détaillant les requêtes SQL en fonction de certains critères. L'affichage des requêtes est effectué par consommation décroissante, ainsi les requêtes les plus gourmandes sont placées en tête de liste.

[2] System Global Area : c'est une zone mémoire qui assure le partage des données entre les utilisateurs. Cette zone contient notamment les structures de données accessible par tous les processus.

```
^LSQL ordered by CPU  DB/Inst: DBXO/DBXO  Snaps: 11-51
-> Resources reported for PL/SQL code includes the resources used by all SQL
   statements called by the code.
-> Total DB CPU (s):             610
-> Captured SQL accounts for  140.0% of Total DB CPU
-> SQL reported below exceeded  1.0% of Total DB CPU

    CPU                  CPU per             Elapsd                     Old
  Time (s)   Executions  Exec (s)  %Total   Time (s)    Buffer Gets  Hash
Value
---------- ------------ ---------- ------ ---------- --------------- ---
    570.33          466       1.22   93.5   18019.29      35,121,130
1862838124
Module: SQL*Plus
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

Le nombre de buffer gets sur cette reqête semble beaucoup trop important par rapport à son nombre d'exécution. Ceci peut expliquer la surconsommation de ressource de la part d'Oracle : lorsqu'une information n'est pas dans le cache d'Oracle, alors un accès disque est réalisé pour lire les données : c'est ce qu'on appelle un buffer get. On peut en conclure que cette requête, ou du moins son plan d'exécution, a besoin d'être optimisé.

```
^LSQL ordered by Elapsed  DB/Inst: DBXO/DBXO  Snaps: 11-51
-> Resources reported for PL/SQL code includes the resources used by all SQL
   statements called by the code.
-> Total DB Time (s):          19,030
-> Captured SQL accounts for  146.2% of Total DB Time
-> SQL reported below exceeded  1.0% of Total DB Time

  Elapsed               Elap per            CPU                        Old
  Time (s)   Executions  Exec (s)  %Total   Time (s)  Physical Reads Hash
Value
---------- ------------ ---------- ------ ---------- --------------- ---
  18019.29          466      38.67   94.7     570.33       1,779,952
1862838124
Module: SQL*Plus
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

```
^LSQL ordered by Gets  DB/Inst: DBXO/DBXO  Snaps: 11-51
-> Resources reported for PL/SQL code includes the resources used by all SQL
   statements called by the code.
-> End Buffer Gets Threshold:     10000 Total Buffer Gets:      34,892,163
-> Captured SQL accounts for  101.1% of Total Buffer Gets
-> SQL reported below exceeded  1.0% of Total Buffer Gets

                                                    CPU      Elapsd     Old
  Buffer Gets    Executions  Gets per Exec  %Total Time (s)  Time (s) Hash
Value
-------------- ------------ -------------- ------ -------- --------- --------
--
     35,121,130          466       75,367.2  100.7   570.33  18019.29
1862838124
Module: SQL*Plus
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

Comme nous pouvons le constater, malgré les différents critères pris en compte, nous avons toujours la même requête qui est la plus consommatrice.

Le plus surprenant est que cette requête est vraiment très simple puisque qu'elle correspond à un simple « select » effectuant une seule jointure.

```
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

Etant donné la simplicité de cette requête, celle-ci devrait être effectuée très rapidement. Cependant dans ce cas précis, il faut 1.8 secondes de CPU ce qui représente à peu près 93% du temps alloué à la base.

c. Administration Web

Après l'expérimentation avec Perfstat, nous avons utilisé l'interface web de la base de données. Celle-ci était toujours accessible à l'url : http://bdd.ig2k.umlv:5503/em/.

Dès le premier coup d'œil, nous avons pu observer que le vumètre « Host CPU » indiquait une charge processeur bien trop élevée (environ 100% en permanence). Nous avons également pu observer que le vumètre « Active Session » indiquait une très importante utilisation des « User IO » (en autre terme des Entrées/Sorties en surnombre, comme nous avions pu l'observer via Perfstat).

```
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

3. Mise en place de la solution

Afin de résoudre les problèmes détectés, nous avons mis un simple mécanisme d'index.

Un index permet des accès rapides aux enregistrements d'une table. Il est créé en utilisant une ou plusieurs colonnes de la table. L'index est généralement utilisé pour les requêtes avec jointures étant appelé un nombre important de fois. Cela définis exactement la requête posant problème sur notre système.

La mesure qui semble la plus adaptée semble donc de poser un index sur :

- WID dans la table BASIC,
- WHATIS dans la table WHATIS,
- Et sur le couple (WID, CLE) dans la table BASIC ;

En effet, ces quatre index permettront au SGBD Oracle de mettre en place un plan d'exécution exploitant au mieux les informations.

4. Rapport à transmettre aux utilisateurs ayant signalé le problème

Mesdames, Messieurs,

Ce rapport fait suite à votre demande d'investigation sur la base de données DBXO située sur le serveur bdd.ig2k.umlv. Celui-ci vous présente les causes du problème, ainsi que les actions préconisées pour le résoudre. Notez que ces opérations peuvent être réalisées dès que vous aurez examiné ce rapport.

1. Causes du dysfonctionnement

Le problème que vous m'avez signalé est réel, et est bien dû au système de gestion de base de données Oracle que vous utilisez. En effet, mes investigations n'ont permis d'incriminer ni une application accédant à la base, ni le système d'exploitation de la machine, ni le réseau.

2. Observations réalisées

Durant mes investigations, j'ai pu analyser pendant une heure un échantillon des requêtes réalisées sur la base de données. Le problème se situe plus particulièrement autour de la requête suivante :

```
SELECT WID,STEX,CLE,TEXTE FROM BASIC, WHATIS WHERE WID=:B1 AND WID=WHATIS
ORDER BY 1,3
```

Bien que celle-ci soit simple, et qu'elle mette en œuvre l'utilisation de paramètres nommés, le plan d'exécution de cette requête n'est pas optimal. En effet, la quantité d'informations stockées dans les tables BASIC et WHATIS sont devenus suffisamment importante pour ne plus pouvoir être stockées entièrement dans le cache du système de gestion de base de données.

Aussi, Oracle effectue de très nombreux accès au disque de stockage de la machine, ce qui coûte environ 1000 fois plus cher qu'un accès au cache. Par conséquence, la sur-utilisation de cette requête provoque une sur-utilisation du stockage, et une importante perte de performances

3. Solution préconisée

Pour remédier à cette situation, je vous conseille de créer les indices suivant sur le tablespace concerné :

- Placer un index sur la colonne WID dans la table BASIC,
- Placer un index sur la colonne WHATIS dans la table WHATIS,
- Et placer un index sur le couple (WID, CLE) dans la table BASIC ;

Cette solution devrait provoquer un gain de performance immédiat dès son application, mais aura des répercutions sur les actions suivantes : le temps nécessaire à l'insertion et la mise à jour d'informations dans ces table sera augmenté, et les opérations de suppression également. Toutefois, étant donné que ces tables sont surtout utilisées en lecture, il apparaît plus efficace de mettre ces index : l'impact en sera minime.

Je me tiens à votre disposition pour toute information complémentaire et en cas de nouvel incident sur ce serveur,

Votre administrateur de base de données,

X.X.

4. Conclusion

Ce TP a été un bon complément aux cours qui avaient été donné préalablement. Il a permis d'une approche très orientée vers des mises en situations réelles rencontrées par un DBA au cours de son travail.

Le TP nous a également permis d'avoir une initiation au monde de la base de données. Il a également permis une idée, bien que succincte, de ce qu'est le travail d'un DBA.

Il est à noter qu'il est dommage de n'avoir eu à disposition qu'une seule base de données pour l'ensemble des étudiants du TP. En effet, cela ne permet pas le recul sur les actions effectuées sur la base ne sachant pas si les résultats observés sont l'effet de nos actions ou de celles d'un autre groupe.

Toutefois, pour avoir pratiqué en entreprise le travail de DBA pour PostgreSQL dans une PME n'ayant pas de DBA attitré, les équipes de développement mettent souvent en cause la base de données lorsque des problèmes sont rencontrés en production, et dans de nombreux cas, à tord. Les outils proposés par Oracle, et avec lesquels nous avons pu nous familiariser, nous ont fait gagner un temps très important par rapport aux outils fournis pour PostgreSQL.

De nombreuses notions sont communes aux deux systèmes (tablespaces, base de données, processus, transactions, utilisateurs…), cependant Oracle s'avère en apparence un meilleur choix quand à la gestion des problèmes de production et à l'administration assistée.