# Optimisation dans
# Le SGBD « Oracle »

# Le réglage des bases de données ?

- Réglage de la configuration initiale en fonction de la charge
- Réglage de l'instance et du SE
  - Identifier les points difficiles et les surmonter
  - Récolter des statistiques sur le fonctionnement
    - des applications
    - de la base et du SE
    - des E/S sur les disques
    - du réseau
  - Découvrir les causes à partir des symptômes et corriger
    - En général : Applications, BD ou matériel
- Réglage des requêtes
  - Souvent écrites à l'intérieur d'outils clients, non optimisées
  - Réglage différent pour OLTP/OLAP
  - Visualiser et contrôler les choix de l'optimiseur

# Les outils Oracle pour le réglage

- Gestionnaire automatique de charge
  - collecte, traite et maintient des stats sur les performances
- Moniteur de diagnostique automatique sur la BD
  - Analyse la charge pour suggérer des problèmes sur la base
- Assistant de réglage SQL
  - Conseils pour l'optimisation des expressions
- Assistant d'accès SQL
  - Conseils sur les index et les vues
- Traceur d'application
  - Pour identifier des surcharges par des applications ou utilisateurs précis
- Sans oublier les alertes serveur (cf la surveillance de la base)

# Concevoir et développer pour la performance

- Eviter absolument les conflits et limites de ressource
- Ne pas penser que l'investissement en matériel va assurer les performances
- Penser en terme de "passage à l'échelle"
  - Comportement linéaire dans la charge de travail
  - Spécificités internet
    - disponibilité 24/24
    - nombre d'accès imprévisible
    - souplesse des requêtes
    - Volatilité et exigence des utilisateurs (7s. d'attente au maximum)
    - Concevoir/développer vite et bien !
  - Causes de mauvaises performances
    - Mauvaise conception, ou mauvaise implémentation
    - Mauvais dimensionnement matériel
    - Limitations logicielles : application, DBMS ou SE

# Concevoir et développer pour la performance (cont.)

- Savoir répondre aux questions suivantes
  - Combien d'utilisateurs à supporter ? très peu, peu à beaucoup, une infinité
  - Quelle mode d'interaction ? Navigateur web ou application cliente personnalisée
  - Où sont les utilisateurs ? (Temps de transfert réseaux)
  - Quelle charge d'accès, combien de données en lecture seule ?
  - Quel est le temps de réponse requis par les utilisateurs ?
  - Quelle disponibilité requise par les utilisateurs ?
  - Mises à jour en temps réel ?
  - Quel taille à prévoir pour les données ?
  - Quelles sont les contraintes budgétaires ?

# Principes pour la conception

- Ne pas faire compliqué quand on peut faire simple
  - Eviter les schémas ou requêtes incompréhensibles (utiliser des vues si besoin)
  - Eviter les superpositions de couches logicielles
- Soigner la modélisation des données pour les parties principales
- Implémenter un schéma en 3NF au moins pour assurer la flexibilité
  - Optimiser avec vues matérialisées, clusters, colonnes calculées
  - Bien organiser les index
- Organiser des campagnes de tests crédibles facilitera le déploiement

# Etapes pour la résolution des problèmes

1. Vérifications préliminaires (avant les problèmes)
   1. Récolter les impressions de base, les projets des utilisateurs
   2. Récolter le maximum de statistiques (SE, DB, applications) lorsque les performances sont bonnes et lorsqu'elles sont mauvaises
   3. Vérifier régulièrement les SE des utilisateurs (matériel, ressources...)
2. Comparer les symptômes avec les "10 erreurs fréquemment commises"
3. Réaliser une modélisation conceptuelle du système lors de l'apparition des symptôme
4. Lister toutes les solutions et les appliquer une à une jusqu'à l'obtention du résultat, ou l'identifiaction des contraintes extérieures conduisant à l'échec.

# Traitement des urgences...

- Bien souvent, un problème doit-être traité dans l'urgence avant une résolution rigoureuse
- Les étapes sont alors "raccourcies" :
  1. Faire l'inventaire des problèmes, des symptômes, des changements récents
- Vérifier l'état du matériel : CPU, disques, mémoire, réseau de chaque tier
- Déterminer si le problème est au niveau du CPU ou de l'attente d'évènement. Utiliser les vues dynamiques sur les performances du catalogue.
- Appliquer des mesures d'urgence pour stabiliser le systèmes : suspendre une application, réduire la charge, tuer un processus...
- vérifier la stabilité du système, récolter des statistiques, et suivre la procédure complète de résolution

# Les 10 erreurs fréquentes selon Oracle

1. Multiplication des connexions à l'instance (une par interaction)
2. Mauvaise utilisation des curseurs et variables liées
   - Les curseurs évitent de recalculer une requête pour usages multiples
   - Les variables liées permettent d'identifier des requêtes similaires
   - Attention au SQL généré dynamiquement par les applications
3. Requêtes SQL inaproprié aux exigences
4. Utilisation de paramètres d'instance non standards
5. Mauvaise répartition des E/S sur les disques
6. Blocages dans les fichiers de reprise
7. Mauvaise gestion des blocks et des segments d'annulation
8. Parcours entier de grandes tables
9. Trop de SQL récursif de la part de SYS (ex. allocation des extents)

# Choix dans la configuration de l'instance

- Considérations initiales
  - Influence de certains paramètres du PFILE
    - Compatible, db_block_size, SGA_TARGET, PROCESSES, SESSIONS, UNDO_MANAGEMENT
  - Gestion automatique des annulations conseillée. Voir V$UNDOSTATS et V$ROLLSTATS
  - Dimension des fichier de reprise : un basculement toutes les 20 minutes...
  - Créer suffisamment de tablespaces
    - Gestion locale recommandée
    - Ne pas oublier les tablespaces temporaires
- Gestion des tables
  - Compresser les tables en lecture seule
  - Récupérer l'espace libre dans les segments
  - Créer les index après le remplissage des tables
  - Laisser Oracle gérer la mémoire pour les tris dans la PGA

# Choix dans la configuration de l'instance (cont.)

- Performances en mode processus serveurs partagés
  - Rappel : un groupe de serveurs pour tous les clients
  - Surveiller la charge des dispacher avec V$DISPACHER et V$DISPACHER_RATE
    - Le taux d'utilisation courrant doit être loin du taux maximal
  - Identifier les blocages des processus serveurs
    - Avec V$QUEUE, pour le temps moyen d'attente des requêtes ou le nombre de serveurs qui tournent actuellement
    - Ajouter si nécessaire des serveurs avec les paramètres d'instance
    - Attention, vérifier si les blocages ne sont pas en mémoire SGA

# Statistiques et diagnostiques automatiques

- Récolter des statistiques est crucial pour résoudre ou devancer les problèmes
  - Concernant la base de données (donnés par Oracle)
    - Les attentes de processus
    - Le temps cumulé de travail de la base (DBTIME) ou de certaines actions
    - Sur le système et les sessions...
  - Concernant le SE (à récupérer avec des outils externes)
    - Sur l'activité des CPU
    - Sur le swapping
    - Les accès disques
    - Le réseau
- Des rapports automatiques de statistiques peuvent être générés en html par ORACLE
- Des diagnostiques et conseils automatiques sont générés à partir des stats
  - Son but est de minimiser DBTIME...

# Gestion de la mémoire

- En général, il est recommandé de laisser Oracle gérer la SGA
  - paramètre SGA_TARGET différent de 0
  - ne gère pas, notamment, le cache de reprise (paramètre LOG_BUFFER)
- Pour une gestion plus fine, on peut récolter des statistiques sur chaque partie de la SGA et gérer sa taille manuellement.
- Surveiller que le cache de reprise n'entraîne pas de ralentissement
  - Stocker les fichiers de logs sur des disques rapides
  - Dans les scripts, ne pas valider à chaque opération
  - Lors de chargement de grands volumes de données, utiliser NOLOGGING
- Utiliser la gestion automatique de la PGA
  - Surveiller et redéfinir si nécessaire

# Gestion des entrées/sorties

- Rechercher la simplicité, tout en assurant les besoins requis. On n'isole des fichiers que pour des raisons de performances ou de pérénnité des données, ou s'il s'agit d'une contrainte de gestion
  - Séparer les fichiers qui requièrent beaucoup d'E/S
  - Une table et son index n'ont pas de raison d'être séparés
  - Essayer tout d'abord de réduire les E/S en optimisant les requêtes...
  - Si le problème vient des fichiers de reprise, les isoler
  - Séparer les fichiers de reprise et les archives
- Penser au mode de gestion automatique des fichiers
- Bien choisir la taille de blocks
  - 8 kilo en général
  - éventuellement plus petit pour un système fortement transactionnel
  - Plus grand en entreposage de données

# Optimiser les performances SQL

# Introduction : Résumé des tâches et outils

- L'un des aspects théorique et pratique les plus importantes en BD
- Trois tâches principales:
  - identifier les charges SQL cruciales pour les performances
  - Vérifier leur plan d'exécution choisi par l'optimiseur
  - Implémenter des améliorations pour améliorer les performances
- Trois directions pour le tuning
  - Réduire la charge : plans d'exécution et index
  - Répartir la charge dans le temps
  - Paralléliser la charge : typiquement en OLAP
- Outils pour le réglage automatique de requêtes
  - Moniteur de diagnostique automatique
  - Moniteur de réglage SQL
  - Moniteur d'accès SQL (index, vues)

# Introduction : Développer des requêtes efficaces

- Tenir à jour les statistiques de l'optimiseur
- Surveiller les plans d'exécution
- Ecrire efficacement les requêtes
  - Utiliser au maximum des AND et = dans les prédicats
  - Ne pas utiliser de fonctions dans les clauses WHERE, en particulier sur des colonnes indexées
  - Décomposer le plus possible les tâches faites par les requêtes
- Choisir le bon connecteur
  - Préférer IN lorsque la sous-requête est la plus sélective
  - Préférer EXISTS dans le cas contraire
- Si nécessaire, maîtriser les choses en utilisant les "HINT"
- Structurer soigneusement les index, supprimer ceux qui sont inutiles
- Limiter les passes sur les données (utiliser le case par exemple)

# Les capacités de réglage automatique de requêtes

- Processus entièrement automatique de réglage des ordres SQL
  - Mode normal : optimiseur classique choisi un "bon" plan
  - Mode tuning : produit des actions possibles pour améliorer
- Processus coûteux : a utiliser uniquement pour les requêtes problématiques
- Base son analyse sur quatre points :
  - les statistiques
  - les profiles SQL = informations avancées sur une requête, dans le dictionnaire
  - les chemins d'accès : index et vues
  - la syntaxe SQL (ex. : UNION ALL plutô que UNION)
- Peut-être utilisé avec SQLAdvisor (graphique ou package PLSQL DBMS_SQLTUNE)
- Possibilité de régler des ensembles de requêtes (SQL Tuning Sets)
  - Un ensemble de requêtes

# Optimiseur de requêtes : ses tâches

- Rappel : SQL est un langage déclaratif
- Optimisation de requête = choix d'un plan d'exécution
  1. Ordre d'évaluation des expressions
  2. transformation des expressions (ex. requêtes imbriquées en jointures)
  3. choix du but, le paramètre à optimiser
  4. Choix des chemains d'accès
  5. ordre de réalisation des jointures
- Les moyens de maîtrise de l'utilisateur :
  - Déterminer le but
  - Récolter des stats
  - Forcer des choix de l'optimiseur avec "HINT"

# Optimiseur de requête : quel objectif ?

- Deux possibilités sous Oracle
  - Minimiser le temps de réponse global (ALL_ROWS, par défaut)
  - Minimiser le temps des n premières réponses (FIRST_ROW)
- n peut être égal à 1, 100 ou 1000
- FIRST_ROWS est à utiliser en cas d'interaction directe avec l'utilisateur
- C'est un paramètre de session
- On peut forcer l'optimiseur avec un HINT
- On peut se baser sur le temps CPU (par défaut) ou les E/S

# Les chemins d'accès du plan d'exécution

- FULL TABLE SCAN : parcours séquentiel complet d'une table
- ROWID SCAN : Accès direct à un ensemble de tuples (après un INDEX SCAN)
- INDEX SCAN : Accès à un index
  1. Unique scan, range scan, full scan, index joins, bitmap joins
- CLUSTER ACCESS : parcours d'un cluster
- SAMPLE TABLE SCAN : accès à un extrait de la base, spécifié dans la requête
- On peut toujours les forcer avec un "HINT"

# Les différents types de jointure

- Jointure imbriquée : double boucle
  - Lorsque peu de lignes doivent être jointes
  - La seconde table est accédée rapidement à partir de l'attribut de jointure
  - O(NxM)
- jointure par hachage : la table la plus petite est "hachée" en mémoire
  - Lorsque la table hachée tient en mémoire
  - Un seul parcours de chaque table
- Jointure par tri fusion : tri puis fusion de chaque opérande
  - Lorsque les sources sont déjà triées
  - La condition de jointure est une inégalité
- Jointure cartésienne : produit cartésien, pas de condition de jointure
- Toutes ces variantes existent en jointure externe.

# Gestion des statistiques

- Ensemble d'information quantitatives sur les données, utilisées par l'optimiseur
  - Sur les tables : Nombre de tuples et de blocks, taille des tuples
  - Sur les colonnes : nb valeurs distinctes et de valeurs NULL, histogramme
  - Sur les index : nb blocks feuilles et niveaux, facteur de regroupement
  - Sur le système : performances E/S et CPU
- Stockées dans le dictionnaire
- Récoltées automatiquement par un processus de fond
  - Pendant une fenêtre temporelle définie (par défaut : 22h-18h + week-end)
  - Cette fenêtre peut-être paramétrée
- On peut lancer manuellement après d'importantes modifications
- Les opérations suivantes peuvent être réalisées sur les stats :
  - restaurer des versions anciennes

# Index : quels colonnes ?

- Quelles colonnes faut-il indexer ?
  - Attributs utilisés fréquemment dans les clause WHERE (jointures ou sélections)
  - L'efficacité augmente avec la sélectivité de l'attribut
  - Automatique pour les clés primaires, unique
  - Eviter d'indexer des colonnes fréquemment modifiées
- Inutile si la clé d'indexation est passée en paramètre d'une fonction
  - Utiliser des index de fonctions
- On peut faire des index composés de plusieurs colonnes
  - Si utilisées ensemble avec une clause AND
  - Placer en premier les attributs les plus fréquemment utilisés
  - Sinon, placer en premier celui sur lequel est ordonnée la table

# Les types d'index

- Par défaut : Oracle utilise des arbres équilibrés (B-arbres)
- Clusters : regroupement de tables dans des blocs communs
  - Autour d'un ensemble de colonnes communes
  - La jointure sur ces colonnes est immédiate
  - A décider à la création des tables
- Index Bitmap
  - Pour des valeurs à faible sélectivité sur des grandes tables
  - Bien adapté à des grandes conjonctions de prédicats
- Tables organisées sur l'index : la table est stockée avec l'index
- On peut indexer les résultats d'une fonction sur une colonne (UPPER, LOWER, ...)
- On peut partitionner un index aussi bien qu'une table

# visualiser les plans d'exécution

- EXPLAIN PLAN
  - Permet de visualiser le plan choisi par l'optimiseur
  - On peut détecter rapidement les points coûteux du plan choisi
  - La requête n'est pas exécutée !
- V$SQL_PLAN
  - Pour voir tous les plans d'exécution des requêtes récemment exécutées
  - On a le plan réel, avec les coûts réels, pas des estimations
- Le plan est stocké dans la table PLAN_TABLE
  - Récupérer les requête d'interrogation de la doc pour une sortie formatée

# Visualiser la charge de travail

1. Fixer les paramètres d'instance pour la collecte de trace
2. Activer la collecte de la trace
   - Produis un fichier avec les statistiques sur toutes les requêtes SQL
3. lancer l'utilitaire TKPROF
   - Pour formater le fichier trace et rendre une sortie lisible
4. Interpréter le fichier généré (voir la doc de TKPROF)
5. Eventuellement, on peut stocker ce résultat dans la base
   - Un script pour cela est fourni par TKPROF