SUPPORT DE COURS ORACLE

CHAPITRE 4

SECURITE DES DONNEES
PAR
LE CONTROLE D'ACCES

SECURITE DES DONNEES PAR LE CONTROLE D'ACCES

SOMMAIRE

Sécurité et contrôle d'accès: les utilisateurs

- Les utilisateurs (users) et leur schèma
- Les profils:

  chaque utilisateur peut voir sa consommation de ressources système contrôlée par un profil spécifique
- Les privilèges d'accès:
  - privilèges système
  - privilèges objets
- Les rôles:

  chaque utilisateur peut voir ses privilèges définis par un rôle particulier.

Création d'un utilisateur de la base

- A chaque création d'un **utilisateur** correspond la création d'un **schéma** de même nom
- Un **mot de passe** est donné à chaque utilisateur
- CREATE USER *utilisateur*
  IDENTIFIED BY *mot_de_passe*/EXTERNALLY
  [DEFAULT TABLESPACE *tablespace*]
  [TEMPORARY TABLESPACE *tablespace*]
  [PROFIL *profil*]
  {QUOTA *entier*/UNLIMITED ON *tablespace*}

Création d'un utilisateur de la base

- Exemple:

```
CREATE USER joël
   IDENTIFIED BY jojo
   DEFAULT TABLESPACE users
   TEMPORARY TABLESPACE temp
   QUOTA 22M ON users
   QUOTA 700K ON user_data;
/*pas besoin de quotas sur temp*/
```


Les profils

- Ensembles nommés de limites ressources
- Affectés à des utilisateurs
  (il ne peut être attribué à chaque utilisateur qu'un seul profil)
- Activés ou désactivés:
  - paramêtre RESOURCE_LIMIT dans init.ora (TRUE ou FALSE)
  - ALTER SYSTEM
    SET RESOURCE_LIMIT = TRUE / FALSE

Les profils

- Avantages:
  - permettent de contrôler les ressources systèmes de chaque utilisateur:
    - interdire des opérations coûteuses
    - s'assurer que les utilisateurs se déconnectent
    - grouper des limites pour des groupes d'utilisateurs
  - sont utiles sur de gros systèmes pour contôler l'utilisation des ressources d'un groupe d'utilisateurs par rapport à un autre
- Inconvénients:
  - charges supplémentaires pour le contrôle de la limite autorisée de chaque utilisateur

Profils et Ressources

Ressource	Description
SESSIONS_PER_USER	nombre maximal de sessions concurrentes autorisées
CPU_PER_SESSION	temps CPU maximal par session en centième de sec.
CPU_PER_CALL	temps CPU maximal pour un appel noyau
CONNECT_TIME	temps de connexion écoulé exprimé en minutes
IDLE_TIME	temps d'inactivité continue exprimé en minutes
LOGICAL_READ_PER_SESSION	nombre maximal de blocs de données lus par session
LOGICAL_READ_PER_CALL	nombre maximal de blocs de données lus par appel
COMPOSITE_LIMIT	coût total des ressources pour une session
PRIVATE_SGA	taille maximale (en K ou M) allouée à la SGA (MTS)

Création, modification et suppression d'un profil

- Création et modification d'un profil:

CREATE / ALTER PROFILE *nom_du_profil*

LIMIT {*ressource entier* / UNLIMITED / DEFAULT}

↳Il existe un profil DEFAULT créé à la création de la base.

- Suppression d'un profil:

DROP PROFILE *nom_du_profil* [CASCADE]

↳CASCADE permet de supprimer le profil pour tous les utilisateurs concernés et de leur affecter le profil DEFAULT.

Création d'un profil

- Exemple:

CREATE PROFILE developpeur_forms LIMIT

SESSIONS_PER_USER	7
CPU_PER_SESSION	UNLIMITED
IDLE_TIME	30
COMPOSITE_LIMIT	1500000;

Quand une limite de ressource est atteinte pour un utilisateur, le SGBD arrête l'exécution de l'opération en cours et annule la transaction.

COMPOSITE_LIMIT RESOURCE COST

- COMPOSITE_LIMIT exprime, pour une session, le coût total des ressources *c_limit*:
  - CPU_PER_SESSION
  - CONNECT_TIME
  - LOGICAL_READ_PER_ SESSION
  - PRIVATE_SGA

- Ce coût est positionné par la commmande:
  ALTER RESSOURCE COST {*c_limit entier*}

COMPOSITE_LIMIT RESOURCE COST

- Exemple:

```
ALTER RESSOURCE COST
  CPU_PER_SESSION        150
  CONNECT_TIME           2;
```

La formule de calcul de ce coût est:

T=150*CPU+2*CON

Le poids par défaut pour une ressource *c_limit* est 0.

Profil par défaut

- Les utilisateurs qui n'ont pas explicitement un profil ont le profil DEFAULT
- Toutes les limites non spécifiées d'un profil ont les valeurs du profil DEFAULT
- Initialement toutes les valeurs par défaut sont illimitées
- Le profil DEFAULT peut être modifié

Vues du dictionnaire et profils

- DBA_USERS: permet d'obtenir des informations sur tous les utilisateurs et leur profil associé
- USER_RESOURCE_LIMITS: décrit les ressources limites de l'utilisateur courant
- DBA_PROFILES:décrit les ressources limites de chaque profil
- RESOURCE_COST: décrit le poids de chaque ressource de COMPOSITE_LIMIT

Les privilèges système

- Un privilège système permet d'exécuter certaines actions touchant la structure de la base:
  - créer une session (une connexion)
  - créer une table ou une séquence etc...:
    - sur son propre schéma
    - sur tout autre schéma,
  - créer un profil, un utilisateur etc...

Attribution de privilèges système

- GRANT *priv._syst. / rôle* {, *priv._syst. / rôle*}
  TO *user / rôle* / PUBLIC {,*user/ rôle* / PUBLIC}
  [WITH ADMIN OPTION]
  - WITH ADMIN OPTION permet aux destinataires de transmettre les privilèges ou les rôles à d'autres utilisateurs ou à d'autres rôles.
  - Retirer les privilèges système à un utilisateur ne se répercute pas sur les autres utilisateurs

Attribution de privilèges système

- Exemple:

GRANT CREATE SESSION,
CREATE TABLE,
EXECUTE ANY PROCEDURE
TO kiki;

Révocation de privilèges système

- REVOKE *priv._syst. / rôle* {, *priv._syst. / rôle*}
  FROM *user/rôle*/PUBLIC{,*user/rôle* /PUBLIC}

Vues du dictionnaire et privilèges système

- SYS.DBA_SYS_PRIVS: décrit tous les privilèges système attribués aux utilisateurs et aux rôles

```
select Grantee,Privilege
       from sys.dba_sys_privs
       where grantee in ('CONNECT','RESOURCE');

GRANTEE                        PRIVILEGE
------------------------------ ----------------------------------------
CONNECT                        ALTER SESSION
CONNECT                        CREATE CLUSTER
CONNECT                        CREATE DATABASE LINK
CONNECT                        CREATE SEQUENCE
CONNECT                        CREATE SESSION
CONNECT                        CREATE SYNONYM
CONNECT                        CREATE TABLE
CONNECT                        CREATE VIEW
RESOURCE                       CREATE CLUSTER
RESOURCE                       CREATE PROCEDURE
RESOURCE                       CREATE SEQUENCE
RESOURCE                       CREATE TABLE
RESOURCE                       CREATE TRIGGER
13 rows selected.
```


Les privilèges objet

- Un privilège objet permet d'exécuter une action particulière sur une table, vue, fonction, séquence, procédure, fonction ou package d'un schéma.

Privilèges Objet	Commentaires
SELECT,DELETE	
INSERT,UPDATE	spécifications de colonnes possibles
EXECUTE	procédure, fonction
ALTER	table, séquence
INDEX	table
REFERENCES	clé étrangère

Attribution de privilèges objet

- GRANT *priv._obj.*[(*liste_de_colonnes*)]
  {,*priv._obj.*[(*liste_de_colonnes*)]} / ALL
  ON [*schéma.*]*objet*
  TO *user* / *rôle* / PUBLIC {,*user*/ *rôle* / PUBLIC}
  WITH GRANT OPTION
  - WITH GRANT OPTION permet aux bénéficiaires des privilèges de transmettre tout ou partie de ces privilèges à d'autres utilisateurs;
  - retirer des privilèges à un utilisateur qui les a reçus avec la clause WITH GRANT OPTION retire en cascade ces privilèges à tous les utilisateurs auxquels il les a transmis.

Attribution de privilèges objet

- Exemples:

  GRANT SELECT, UPDATE(adr,ville) ON fournisseurs
  TO PUBLIC;

  GRANT ALL ON entete TO joël, robert, marcel;

  GRANT EXECUTE ON conception TO eric, cecilia
  WITH GRANT OPTION;

Révocation de privilèges objet

- REVOKE *priv._obj.* {, *priv._syst.*} / ALL
  ON [*schèma.*]*objet*
  FROM *user/rôle* {,*user/rôle*} / PUBLIC
  [CASCADE CONSTRAINTS]
  - CASCADE CONSTRAINTS permet de supprimer toutes les contraintes d'intégrité référentielle définies sur les objets pour lesquels on demande le retrait de privilèges.

Vues du dictionnaire et privilèges objets

- USER_TAB_PRIVS [_MADE / _RECD] décrit les privilèges objets donnés ou reçus directement
- USER_COL_PRIVS [_MADE / _RECD] décrit les colonnes spécifiées dans les privilèges
- DBA _TAB_PRIVS décrit tous les privilèges sur tous les objets
- DBA _COL_PRIVS décrit toutes les colonnes spécifiées dans les privilèges

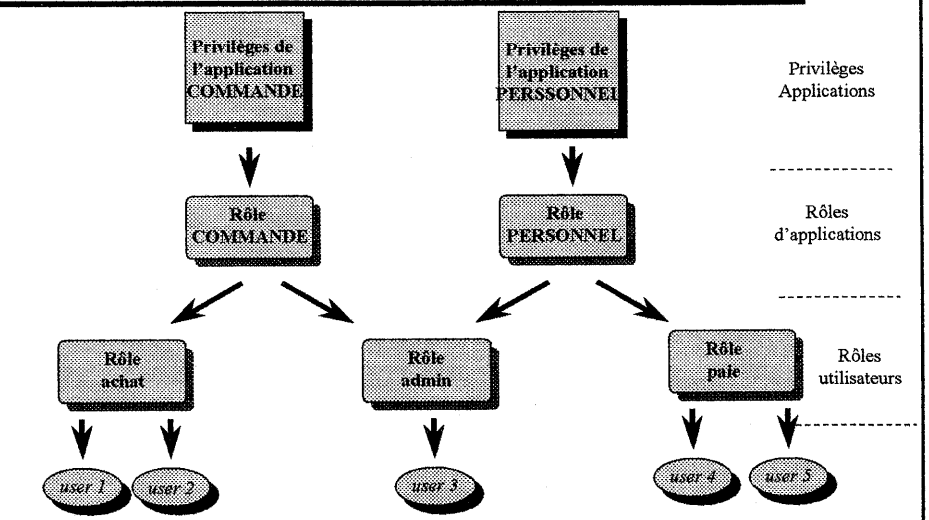
Vues du dictionnaire et objets

- ALL_TAB_PRIVS [_MADE / _RECD] décrit les privilèges objets donnés ou reçus directement, via un rôle ou PUBLIC
- ALL_TAB_PRIVS_MADE décrit les privilèges objets donnés via un rôle GRANT OPTION
- ALL_TAB_PRIVS_RECD décrit les privilèges objets reçus directement, via un rôle ou PUBLIC
- description sur les colonnes en remplaçant dans les noms de vues _TAB par _COL

Les rôles

- Ensemble nommé de privilèges:
  - composé de privilèges système ou/et objet
  - attribué à un utilisateur ou/et à un autre rôle
  - activé ou désactivé pour chaque utilisateur
  - pouvant nécessiter une autorisation
  - n'appartenant à aucun schéma
- Avantage:
  - centralise la gestion des privilèges
  - différencie les privilèges nécessaires aux applicatifs (rôle d'application) et ceux propres aux utilisateurs (rôle utilisateur)

Utilisation des rôles



Création, modification et suppression d'un rôle

- Création et modification d'un rôle:

CREATE / ALTER ROLE *rôle*
[NOT IDENTIFIED
/ IDENTIFIED BY *mot_de_passe* / EXTERNALLY]

- Suppression d'un rôle:

DROP ROLE *rôle*

- Exemple:

CREATE ROLE admin
IDENTIFIED BY adios

Activation d'un rôle

- SET ROLE *rôle* [IDENTIFIED BY *mot_de_passe*]
  {,*rôle* [IDENTIFIED BY *mot de passe*]}
  /ALL [EXCEPT *rôle* {, *rôle* }]
  /NONE
  - NONE: désactive tous les rôles de la session courante
- Exemple:

  SET ROLE appli_partie_1; .....
  SET ROLE appli_partie_2; .....
  SET ROLE ALL EXCEPT appli_partie_1 ;

Vues du dictionnaire et rôles

- SESSION_PRIVS: privilèges actifs de l'utilisateur courant
- SESSION_ROLES: rôles actifs de l'utilisateur ...
- USER_ROLE_PRIVS: rôles de l'utilisateur ...
- DBA_ROLES: tous les rôles
- DBA_SYS_PRIVS: privilèges système attribués aux utilisateurs et aux rôles
- ROLE_ROLE_PRIVS: informations sur les rôles attribués à d'autres rôles

Création d'un utilisateur de la base

- CREATE USER *utilisateur*
  IDENTIFIED BY *mot_de_passe*/EXTERNALLY
  [DEFAULT TABLESPACE *tablespace*]
  [TEMPORARY TABLESPACE *tablespace*]
  [PROFIL *profil*]
  {QUOTA *entier*/UNLIMITED ON *tablespace*}
- Exemple:
  CREATE USER joël IDENTIFIED BY jojo
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  PROFILE developpeur_forms
  QUOTA 22M ON users QUOTA 700K ON user_data;

Modification d'un utilisateur

- ALTER USER *utilisateur*
  [IDENTIFIED BY *mot_de_passe*/EXTERNALLY]
  [DEFAULT TABLESPACE *tablespace*]
  [TEMPORARY TABLESPACE *tablespace*]
  [PROFIL *profil*]
  {QUOTA *entier*/UNLIMITED ON *tablespace*}
  [DEFAULT ROLE *rôle*{,*rôle*}
  /ALL[EXCEPT *rôle*] /NONE]
- Exemple:
  ALTER USER joël DEFAULT RÔLE admin;

Suppression d'un utilisateur

- DROP USER *utilisateur* [CASCADE]
  - CASCADE: tous les objets du schéma associé sont supprimés.

Suppression d'une session utilisateur

- ALTER SYSTEM KILL SESSION 'sid,n°série'
  - sid: numéro de session utilisateur
  - n°série:numéro de série de la session

    sont contenus dans la vue V$SESSION
- Exemple:

```
SELECT sid,serial#,username
  FROM v$session WHERE username ='joël';
        12     97     joël
ALTER SYSTEM KILL SESSION '12,97'
```

Vues du dictionnaire et utilisateurs

- USER_USERS: permet d'obtenir des informations sur l'utilisateur courant
- ALL_USERS et DBA_USERS: permettent d'obtenir des informations sur tous les utilisateurs
- USER_TS_QUOTAS: donne les quotas par tablespace pour l'utilisateur courant
- DBA_TS_QUOTAS: donne les quotas par tablespace pour tous les utilisateurs

Exercices: gestion des utilisateurs

1. Facultatif: changer votre mot de passe (sans l'oublier aussitôt après!!!)
2. Créer un utilisateur *Vincent(n)* ayant pour mot de passe *vinc*. Le tablespace temporaire de *Vincent(n)* sera TEMP.
3. Tenter de se connecter à *Vincent(n)*. Que se passe-t-il? Pourquoi?
4. Créer un utilisateur *Theo(n)* ayant pour mot de passe *theo*. Le tablespace temporaire de *Vincent(n)* sera TEMP et son tablespace par défaut sera USER_DATA limité à 3 M octets.
5. Interroger la vue V$SESSION pour déterminer le SERIAL# et le SID de votre session; écrire la commande qui devrait être utilisée pour mettre fin à votre connexion à la base de données. Qu'est il arrivé et pourquoi?
6. Créer un profil *une_deux(n)* qui autorise une seule connexion et 2 minutes d'inactivité. Attribuer ce profil à *Theo(n)*.
7. Attribuer à *Vincent* le privilège nécessaire pour se connecter à la base. Se connecter à *Vincent* et interroger les vues SESSION_PRIVS et USER_SYS_PRIVS.
8. Attribuer à *Theo(n)* les privilèges nécessaires pour se connecter à la base et créer des tables.
9. Attribuer à *Theo(n)* les privilèges nécessaires pour créer la table *dept* à partir de celle qui vous appartient. Se connecter à *Theo(n)*, créer la table *dept* et visualiser la vue USER_OBJECTS. Visualiser la vue USER_OBJECTS de votre compte.
10. *Theo(n)* autorise *Vincent* à lire sa table *dept*. Vérifier à l'aide de la vue USER_TAB_PRIVS.
11. *Theo(n)* autorise *Vincent(n)* à modifier la colonne ville de sa table *dept*. Se connecter à *Vincent(n)* et interroger les vues USER_TAB_PRIVS et USER_COL_PRIVS. *Vincent(n)* transfert tous les départements de Paris à Rouen et change le nom du département *ventes* en *achats*. Que se passe-t-il? Faire un rollback.
12. Créer un rôle *developpeur(n)* qui autorise les utilisateurs à créer des tables, vues, séquences et synonymes.
13. Attribuer ce rôle a *Theo(n)*. Tester.

SUPPORT DE COURS ORACLE

CHAPITRE 5

STRUCTURES DE STOCKAGE, INDEX ET CLUSTERS

- Le choix d'un type de stockage adapté peut réduire les temps d'accès aux informations.
- Certains types de stockage sont meilleurs pour certaines opérations.

La structure séquentielle (heap):

C'est généralement la structure par défaut.

Les interrogations se font sur toute la table, rendant cette structure inefficace pour la recherche sur des tables de taille importante.

Les ajouts se font à la fin de la table, rendant cette structure la plus rapide quand un grand nombre de lignes est ajouté (surtout quand la table est vide au départ).

Les clés ne peuvent pas être utilisées dans la structure séquentielle.

Les lignes dupliquées ne sont pas gérées (c.a.d. pas otées contrairement aux autres structures).

Les effacements laissent des trous.

Utilisation : Cette structure est utilisée plus particulièrement pour le chargement des tables ou pour de petites tables.

La structure hash:

L'algorithme qui permet d'ordonner les données, est basé sur le nombre de lignes de la table et le nombre de lignes que peut contenir une page. En conséquence un nombre de pages, pour les pages dites *pages principales*, est choisi à la création de la structure . Chaque ligne est placée sur une page principale en fonction de la valeur *'hashée'* de sa clé. Le nombre de lignes sur chaque page varie en fonction des données.

Utilisation : Cette structure est très efficace pour attaquer une valeur précise. Si la clé est constituée de plusieurs colonnes, une valeur pour la clé entière doit être spécifiée afin que la structure soit utilisée , lors d'une requête.

Exemple : Soit une table *employes* conservant l'âge des employés.

Supposons que l'algorithme de hash coding soit :

numéro de page principale =

clé modulo (nombre de pages principales)

avec pour clé : l'âge,

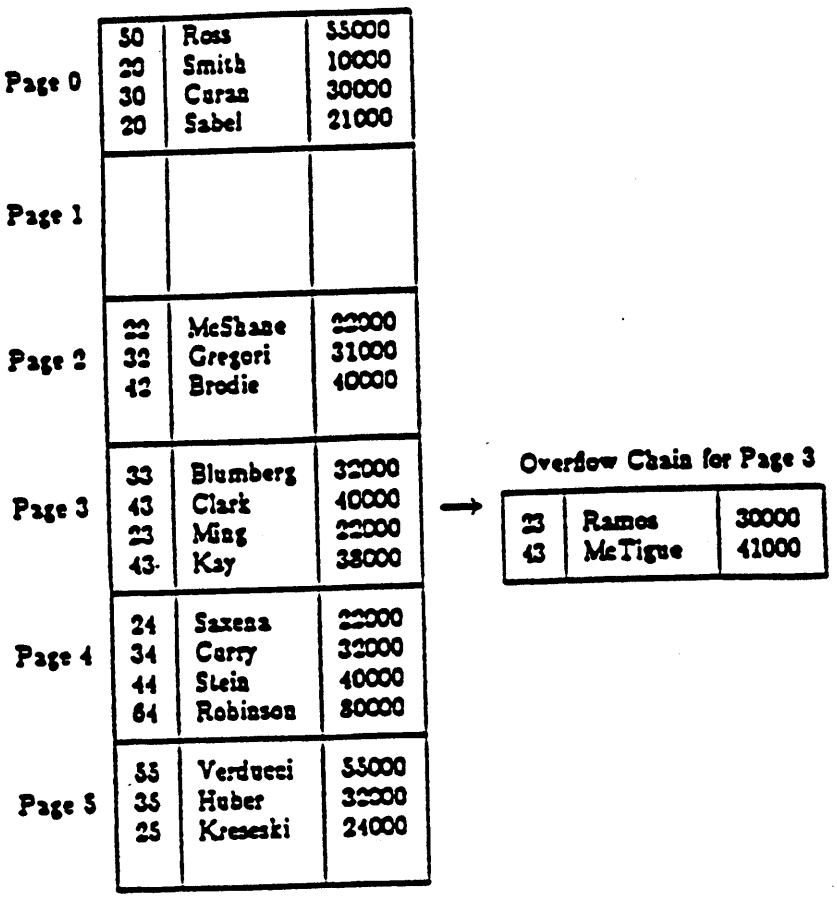
et nombre de pages principales :10.

La requête : select * from employes where age = 43;

recherche au travers de la page 3 et de sa page de débordement,

la requête : select * from employes where age > 43;

effectue une recherche sur toute la table.



La structure isam:

Ici, les tables sont physiquement triées sur les valeurs de la colonne ayant servi à créer la structure. Ce type de stockage permet des recherches sur des plages de valeurs. Le nombre de pages d'index et de pages principales est statique, et fixée à la création du type isam. Les clés sont triées de gauche à droite et l'utilisateur doit spécifier la partie la plus à gauche avant de spécifier la partie droite, mais contrairement aux tables 'hashées' la partie droite peut être omise.

Utilisation : Cette structure est utilisée quand la table s'étoffe lentement, quand la clé est importante (données de la clé non dupliquées contrairement au btree), quand la table est suffisament petite afin de pouvoir reconstruire l'index, quand la clé n'est pas forcément attaquée avec une valeur exacte.

Exemple 1: Soit la table *employes* conservant l'âge des employés, et une clé construite sur le nom et l'âge.
Les requêtes:

```
select * from employes
   where nom='Dubois' and age>43;
select * from employes
   where nom like 'D%';
```

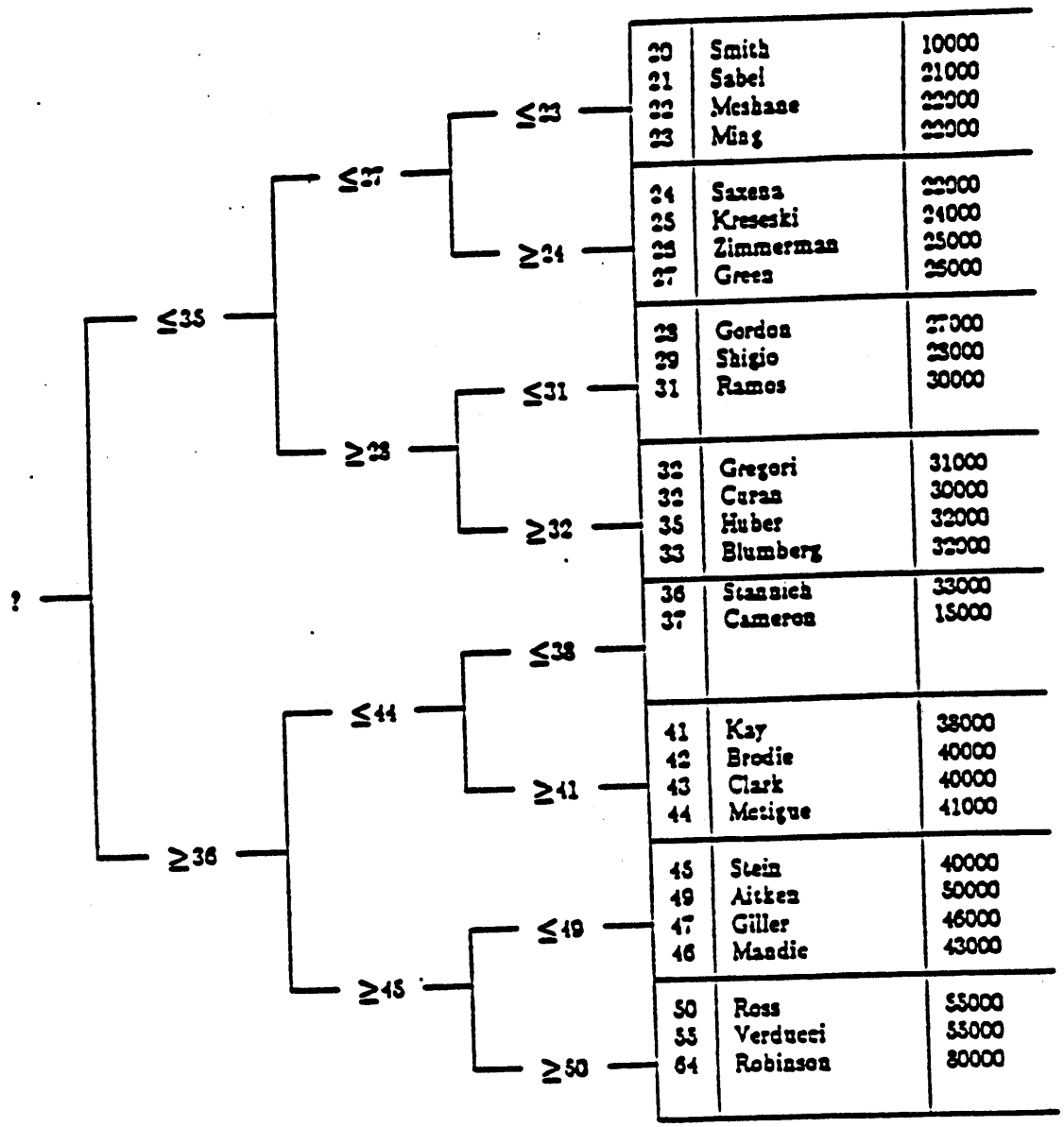
utiliseront la structure,

les requêtes:

```
select * from employes
   where nom like '%bois%' and age>43;
select * from employes where age>43;
```

n'utiliseront pas la structure.

Exemple 2: Soit la table *employes* conservant l'âge des employés et une clé sur l'âge;



La structure btree:

Les caractéristiques sont les mêmes que pour l'isam, à la différence que ici, l'index est dynamique.

Structure général du btree:

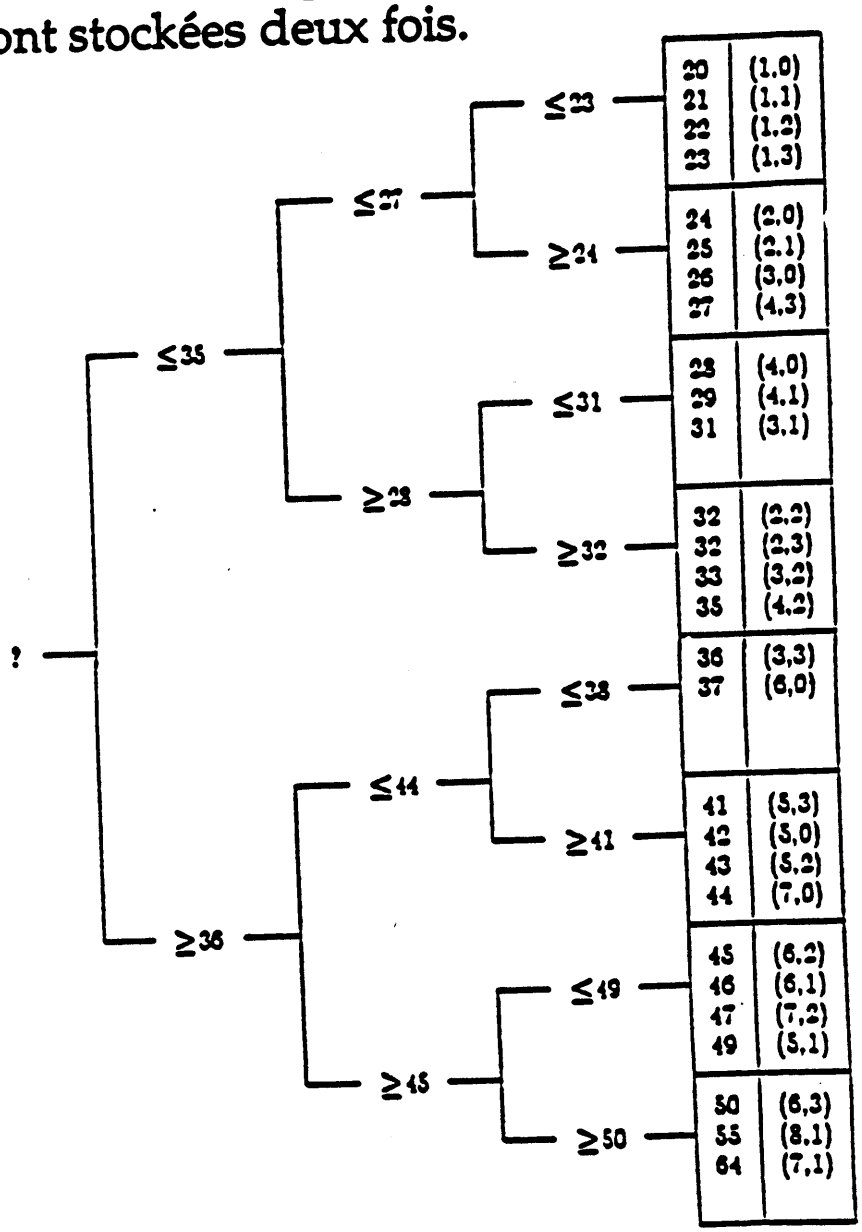
###la page d'entête contrôle l'index, les feuilles et les pages de données.

###l'index du btree pointe sur les feuilles, tandis que l'index de l'isam pointe sur les pages de données. Une page d'index pointe soit sur d'autres pages d'index, soit sur des feuilles.

###les feuilles contiennent les clés et pointent vers les pages de données, dites pages de données associées. Les données sur les pages de données ne sont pas déplacées

###l'espace libéré par des effacements est réutilisé.

Utilisation : Cette structure est utilisée quand la table s'étoffe rapidement et est trop grande pour que l'on puisse changer la structure. On évitera de l'utiliser si la clé est importante car les données de la clé sont stockées deux fois.



Data pages:

Smith	20
Sabel	21
McShane	22
Ming	23

1

Saxena	24
Kreseski	25
Gregori	32
Curran	32

2

Zimmermann	26
Ramos	31
Blumberg	33
Stanwich	36

3

Gordon	28
Shigio	29
Huber	35
Green	27

4

Brodie	42
Aiken	49
Clark	43
Kay	41

5

Sous Ingres:

Il est possible de modifier à tout moment la structure de stockage d'une table afin:

- que la structure de stockage mise en place réponde au mieux en termes de performances aux requètes des applicatifs
- de suivre l'évolution des données en termes de volume et de distribution.

✎*Attention*: la réorganisation d'une table détruit tous les index de cette table.

Modification de la structure de stockage:

MODIFY nom de table ou d'index
    TO structure de stockage [UNIQUE]
    ON colonne[ASC/DESC] {,colonne[ASC/DESC]}
    [WITH clause {,clause}]

structure de stockage : HEAP, HASH, ISAM, BTREE
clause : MINPAGES, MAXPAGES, FILLFACTOR, LEAFFILL, NONLEAFFILL

Sous Oracle:

Création et modification d'un index:

```
CREATE INDEX [UNIQUE / BITMAP] [schéma.]index
      ON [schéma.]table (colonne [ASC/DESC]
                        {,colonne [ASC/DESC]}
            /CLUSTER [schéma.]cluster
      [TABLESPACE nom_de_tablespace]
      [PCTFREE entier]
```

Exemple:

```
CREATE INDEX fou_vil
      ON fournisseurs (ville)
      TABLESPACE tbs_ndx
```

Les Clusters

Définition: Un **cluster** est un regroupement **physique** de 2 ou plusieurs tables (jusqu'à 16) autour d'une ou de plusieurs colonnes.

Avantages: Amélioration des performances lors des opérations de jointure. Utilisation transparente des clusters pour les applicatifs et les utilisateurs.

Inconvénients: Dégradation des performances lors de recherche sur des tables simples.

Avant de créer un cluster:
- connaître les requêtes de consultation de l'application
- déterminer les colonnes communes à ces tables afin de construire la clé du cluster.

Colonnes d'une clé de cluster:
- contiennent un grand éventail de valeurs
- sont utilisées en jointure de plusieurs tables
- ne sont pas fréquemment mises à jour
- ne peuvent pas être de type LONG ou LONG RAW.

Types de clusters:
- les clusters indexés
- les clusters à hashage

Les Clusters indexés

Un index est créé sur les colonnes de la clé du cluster, après la création du cluster, avant l'insertion des données (contrairement à une table les données ne sont pas accessibles sans l'index du cluster).

Création et modification d'un cluster:

```
CREATE / ALTER  CLUSTER [schéma.]cluster
      (colonne type {,colonne type})
      [SIZE entier [K/M]]
      [TABLESPACE nom_du_tablespace]
      [INDEX]
```

- SIZE entier [K/M] : spécifie l'espace en octets, Ko. ou Mo. permettant de stocker toutes les lignes d'une valeur de clé particulière
- INDEX : crée un cluster index

Suppression d'un cluster (quelque soit le type):

```
DROP CLUSTER [schéma.]cluster
      INCLUDING TABLES
      CASCADE CONSTRAINTS
```

- INCLUDING TABLES: supprime les tables du cluster ou les supprimer au préalabe (sinon erreur)
- CASCADE CONSTRAINTS : supprime les contraintes d'intégrité référentielles (sinon erreur possible)

Les Clusters à hashage

Une fonction de hashage est appliquée à une colonne ou à une clé composée d'une table. On utilise ce type de cluster quand les donnnées, en général, ne varient pas.
L'optimisation des performances est l'objectif essentiel.

Création et modification d'un cluster:

CREATE / ALTER CLUSTER [schéma.]cluster
(colonne type {,colonne type})
[HASH IS colonne]
HASHKEYS entier /* Oracle arrondit au nombre premier supérieur */
SIZE entier [K/M]
TABLESPACE nom_du_tablespace

- HASHKEYS : permet de créer un cluster à hashage (par défaut un cluster indexé) et l'entier spécifie le nombre de valeurs hash (fixé à la création).
- HASH IS colonne : spécifie une colonne de type number utilisée pour les valeurs de hashage
- SIZE entier [K/M] : spécifie l'espace permettant de stocker toutes les lignes ayant la même valeur de clé de hashage.

Exercices:

1. Quel est l'espace moyen utiltisé par une ligne de la table *employes*? (la fonction vsize(*expression*) retourne le nombre d'octects nécessaires pour stocker *expression*).
2. Même question pour dept.
3. A partir des réponses aux questions 1 et 2, mettre les tables employes et dept en cluster et choisir un bon paramêtre pour SIZE.
4. Créer une copie de la table employes, appelée emp_bis, qui permettra d'accéder aux données par une fonction de hashage sur la colonne empno.