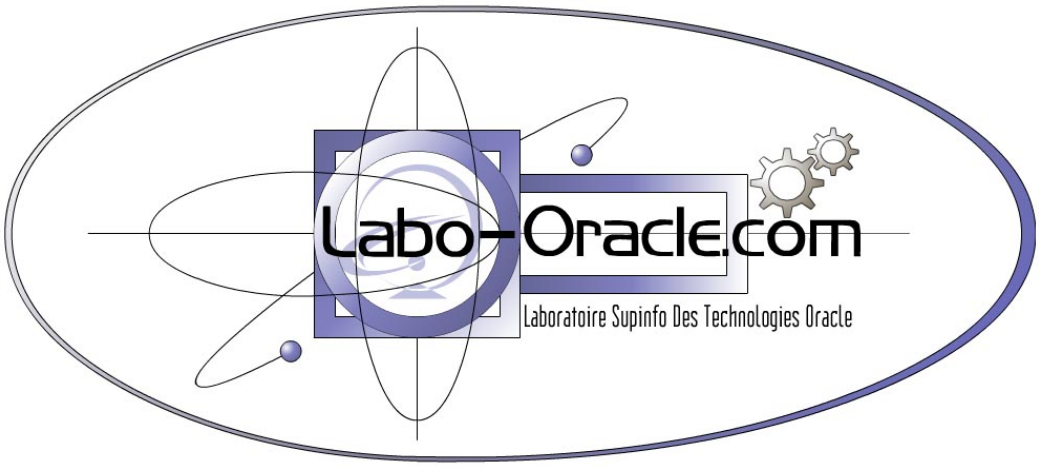


Module n°3

GESTION DES STRUCTURES DE STOCKAGE

1Z0-001

Auteur : Aurélie Vuaroqueaux
Version 1.3 – 7 août 2003
Nombre de pages : 37

Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

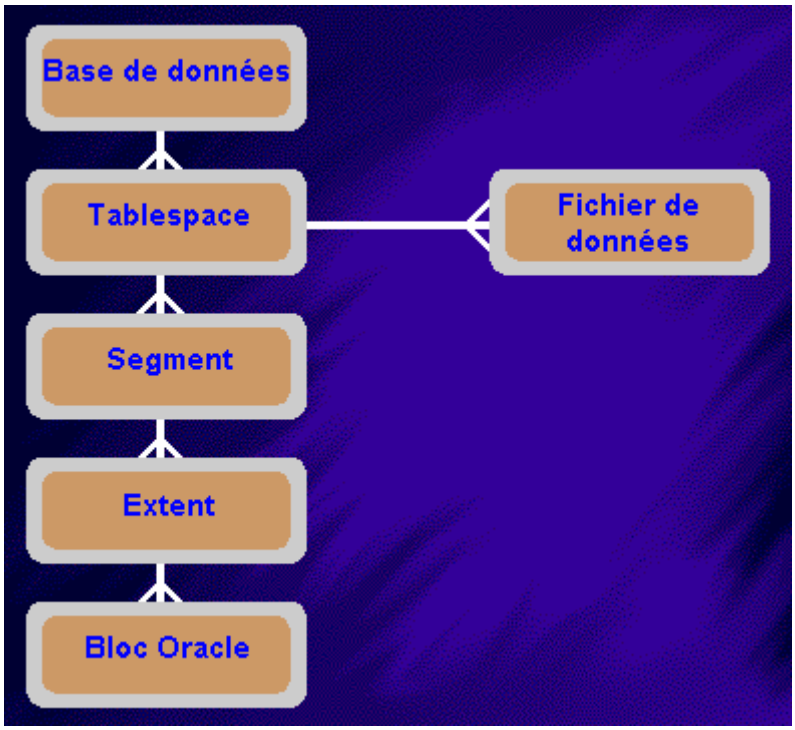
Table des matières

1. Structures de stockage

1.1. Segments

1.1.1. Hiérarchie de stockage d'une base de données

Une base de données Oracle stocke des données qui peuvent être sélectionnées.
La hiérarchie de stockage d'une base de données qui contient différents composants est la suivante :



La base de données, premier composant dans la hiérarchie de stockage, est divisé logiquement en tablespaces.

Le tablespace est le deuxième composant dans la hiérarchie de stockage. Il permet de regrouper des structures logiques liées. Par exemple, les tablespaces peuvent regrouper tous les objets d'une application pour simplifier certaines tâches d'administration.
Chaque tablespaces contient un ou plusieurs fichiers de données.

Les fichiers de données sont les structures physiques qui stockent les données.
La somme des tailles des fichiers de données d'un tablespace correspond à la capacité de stockage de tous les tablespaces. La somme des capacités de stockage de tous les tablespace d'une base de données donne la capacité de stockage totale de cette base de données.

Un tablespace peut contenir un ou plusieurs segments.
Un segment correspond à l'espace utilisé par une structure logique, par exemple une table ou un index. Lors de sa création, un segment contient au moins un extent.

Le quatrième composant de la hiérarchie de stockage est l'extent. Il s'agit d'un ensemble de blocs contigus permettant de stocker un certain type d'information. Des extents sont ajoutés lorsqu'on segment nécessite davantage d'espace.

Le dernier composant de la hiérarchie de stockage est le bloc Oracle. Il s'agit de la plus petite unité d'entrée/sortie. Lorsque des données doivent être extraites du disque, le serveur Oracle utilise un ou plusieurs blocs Oracle. La taille d'un bloc Oracle doit être un multiple de la taille d'un bloc du système d'exploitation.
Un bloc de données correspond à un nombre spécifique d'octets d'espace de base de données physique sur le disque. Lors de la création de chaque base de données Oracle, une taille de bloc de données est définie.

1.1.2. Types de segments

Des segments sont créés dans la base de données pour stocker, extraire et gérer des données.

Un segment est un objet occupant de l'espace dans la base de données. Il est constitué d'un ou plusieurs extents.
Les différents types de segments sont les suivants :

- Table
- Partition de table
- Cluster
- Table organisée en index
- Segment LOB
- Table imbriquée
- Index
- Partition d'index
- Index lob
- Rollback segment
- Segment temporaire
- Segment de démarrage

La table, également appelée table non partitionnée, constitue la méthode de stockage de données la plus courante dans une base de données.
Les données d'une table sont stockées de façon aléatoire et l'administrateur de base de données contrôle peu l'emplacement des lignes dans les blocs d'une table. Toutes les données d'une table non partitionnée doivent être stockées dans un tablespace.

Dans une table partitionnée, les données d'une table volumineuse peuvent être réparties entre plusieurs tablespaces dans des unités logiques appelées partitions. Une table peut être partitionnée en fonction d'une plage de valeurs de clé.
Si une table est partitionnée, chacune des partitions est stockée dans un segment distinct. Les paramètres de stockage peuvent être définis de façon à contrôler chaque partition de table séparément. Cela permet d'accroître l'évolutivité et la disponibilité des données.

Un cluster peut contenir une ou plusieurs tables. Les lignes de ces tables sont stockées en fonction de leur valeur de colonne clé. Toutes les tables d'un cluster sont stockées dans un seul segment et partagent les mêmes caractéristiques de stockage.

Dans une table organisée en index, les données sont stockées dans l'index en fonction de leur valeur clé. Toutes les données d'une table organisée en index peuvent être extraites directement de l'arborescence de l'index. La totalité de la table organisée en index est stockée dans un seul segment.

Une ou plusieurs colonnes d'une table peuvent servir au stockage des objets volumineux (LOB, large objects), tels que les images. Le serveur Oracle stocke ses LOBs dans des segments distincts appelés segment LOB. L'emplacement des données LOB est indiqué par un pointeur dans la table.

Dans Oracle, il est possible de créer une table contenant une colonne dont le type de données est une autre table. On parle alors de table imbriquée. Une table imbriquée est toujours stockée en tant que segment distinct.

Un seul segment est alloué à chaque index d'une base de données Oracle. Ce segment contient toutes les entrées de l'index considéré. Par exemple, si une table a trois index, trois segments d'index sont utilisés. Un index permet d'extraire l'emplacement des lignes d'une table pour un ensemble de valeurs de clé donné.

Un index peut être partitionné et réparti entre plusieurs tablespaces. Chaque partition de l'index correspond à un segment et doit être stockée dans un seul et même tablespace. Une partition d'index permet de réduire la contention en répartissant les entrées/soties sur l'index.

Un index LOB est créé implicitement lors de la création d'un segment LOB. Un index LOB est utilisé par le serveur Oracle pour extraire ou manipuler des données LOB. Vous pouvez définir les caractéristiques de stockage de l'index LOB.

Un rollback segment est utilisé par une transaction qui apporte des modifications à une base de données. Avant toute modification d'un bloc de données ou d'un bloc d'index, l'ancienne valeur est stockée dans le rollback segment. Cela permet à l'utilisateur d'annuler les modifications apportées à la table de données, si nécessaire.

Lorsqu'un utilisateur exécute une commande qui nécessite un tri et que le serveur Oracle ne peut pas effectuer ce tri dans la mémoire, le serveur Oracle alloue à l'utilisateur un segment temporaire dans un tablespace. Ce segment temporaire sert à stocker les résultats intermédiaires lors de l'exécution de la commande.

Un segment de démarrage, également appelé segment en mémoire cache, est créé par le script SQL.BSQ lors de la création de la base de données. Ce segment permet d'initialiser le cache du dictionnaire de données lorsque la base de données est ouverte par une instance Oracle.
Le segment de démarrage ne peut pas être interrogé ni mis à jour. Il ne nécessite aucune maintenance de la part de l'administrateur de base de données.

1.2.Gestion des extents

1.2.1. Contrôle de l'allocation d'extents

Le serveur Oracle alloue un extent à un segment lorsque ce segment est plein. Le processus de contrôle de l'allocation d'extents à un segment est défini par un paramètre de stockage au niveau du segment.
Les paramètres de stockages utilisés pour l'allocation d'extents sont soumis à certaines règles de priorité. Tout paramètre de stockage spécifié au niveau du segment (sauf le paramètre de tablespace MINIMUM EXTENT) annule l'option correspondante définie au niveau du tablespace (la cause MINIMUM EXTENT n'existe pas pour les segments).

Ensuite, si les paramètres de stockage ne sont pas définis explicitement au niveau du segment, ils héritent les valeurs par défaut des paramètres de stockage définis au niveau du tablespace.

Si l'option DEFAULT STORAGE n'est pas spécifiée pour le tablespace, celui-ci hérite les valeurs pas défaut définies pour cette version d'Oracle.

Un certain nombre de facteurs influencent l'allocation d'espace de stockage :

- Si les paramètres de stockage sont modifiés, les nouvelles options définies s'appliquent uniquement aux nouveaux extents.
- Certains paramètres ne peuvent pas être définis au niveau du tablespace. Ces paramètres peuvent être définis uniquement au niveau du segment.
- La taille d'extent minimale spécifiée pour un tablespace s'applique à tous les extents qui seront alloués aux segments de ce tablespace.

1.2.2. Fusion de l'espace libre

Un extent est une unité logique permettant d'allouer de l'espace à une base de données. Il est constitué d'un ensemble de blocs de données contigus. Un segment comporte un ou plusieurs extents.
Des extents sont alloués à un segment, ou au contraire libérés, sous certaines conditions. Ces conditions sont les suivantes :

Allocation ou libération d'extents	
Un extent est alloué lorsque le segment est :	**Un extent est libéré lorsque le segment est :**
o Créé o Etendu o Modifié	o Supprimé o Modifié o Vidé o Redimensionné automatiquement

Lors de sa création, un segment se voit attribuer de l'espace provenant des extents libres d'un tablespace. L'espace contigu utilisé par un segment est appelé extent utilisé. Lorsqu'un segment libère de l'espace, les extents libérés sont ajoutés au pool d'extents libres du tablespace.

De l'espace contigu peut être libéré lorsque plusieurs extents sont libérés au sein d'un même tablespace. Cela peut notamment se produire quand deux tables sont supprimées. A l'issue d'une libération d'espace, il peut exister deux extents libres contigus.

Des extents contigus peuvent être fusionnés pour former un seul extent. Cette fusion peut intervenir sous certaines conditions :

- Lorsque le processus System Monitor (SMON) démarre une transaction d'espace pour fusionner deux extents libres adjacents.
- Lorsque le serveur Oracle doit allouer un extent qui nécessite de l'espace provenant de plusieurs extents libres adjacents.
- Lorsque l'administrateur de base de données en fait la demande.

Des extents d'un tablespace peuvent être fusionnés à l'aide d'Oracle SQL Worksheet ou SQL*Plus. L'administrateur peut interroger la vue DBA_FREE_SPACE_COALESCED pour savoir s'il a besoin de fusionner des extents libres dans un tablespace.

L'administrateur permet de fusionner tout l'espace libre contenu dans les extents d'un tablespace pour former des extents contigus plus grands.
La commande utilisée pour effectuer cette opération est :

```
ALTER TABLASPACE tablespace
COALESCE ;
```

Exemple :

Fusionner l'espace libre qui se trouve dans les extents du tablespace RBS.

```
SQl> ALTER TABLESPACE rbs COALESCE ;
Tablespace altered.
```

1.2.3. Tendance à la fragmentation : objets

Les différents types de segments n'ont pas la même tendance à la fragmentation. Pour limiter le gaspillage d'espace, il est conseillé de placer les différents types de segments dans des tablespaces distinctes.

La structure conseillée pour les tablespaces, les objets qu'ils contiennent et leur tendance à la fragmentation sont affichées dans le tableau ci-dessous.

Organisation des tablespaces en fonction de leur tendance à la fragmentation		
Tablespace	**Utilisation**	**Fragmentation**
SYSTEM	Dictionnaire de données	Nulle
TOOLS	Applications	Très faible
DATAn	Segments de données	Faible
INDEXn	Segments d'index	Faible
RBSn	Rollback segments	Elevée
TEMPn	Segments temporaires	Très élevée

Les tablespaces à destination des segments temporaires présentent une fragmentation élevée si ces tablespaces sont de type PERMANENT. Dans le cas où ils sont de type TEMPORARY, il n'y a plus de fragmentation entre extents.

Les objets du dictionnaire de données ne sont jamais supprimés ni vidés, à l'exception de la table d'audit. Leur tendance à la fragmentation est nulle. Par conséquent, ils ne risquent pas de fragmenter le tablespace.

L'espace utilisé pour les repository des applications, comme Oracle Entreprise Manager et Designer/2000, peut être libéré uniquement lors de la réorganisation de ces structures. Etant donné que ces tables sont rarement réorganisées, ces objets ont une tendance à la fragmentation très faible.

Les segments de données et d'index utilisés pour les applications écrites par l'utilisateur peuvent nécessiter des réorganisations plus fréquentes que les repository. Par conséquent, ils ont une tendance à la fragmentation plus élevée que les repository d'applications.

Les rollback segment sont susceptibles de provoquer une fragmentation dans un système ayant une forte activité de mise à jour, car ils peuvent libérer des segments automatiquement.

Les segments temporaires dans des tablespaces permanents peuvent libérer de l'espace assez fréquemment. Par conséquent, leur tendance à la fragmentation est très élevée. Ils doivent donc être placés dans des tablespaces distinct. Les tablespaces à destination des segments temporaires présentent une fragmentation élevée si ces tablespaces sont de type PERMANENT. Dans le cas où ils sont de types TEMPORARY, il n'y a plus de fragmentation entre extents.

1.3.Espace dans un bloc

1.3.1. Contenu d'un bloc de données

Oracle stocke les données dans des blocs de données.
Un bloc de données a diverses caractéristiques :

- Plus petite unité d'E/S
- comprend un ou plusieurs blocs OS (Operating System)

- Défini par la propriété DB_BLOCK_SIZE
- Défini lors de la création de la base de données.

Un bloc de données est composé de trois éléments :
- l'en-tête du bloc,
- l'espace libre et
- l'espace des données.

L'en-tête contient l'adresse du bloc de données, le répertoire des tables, le répertoire des lignes et les entrées de transaction. Les entrées de transaction sont utilisées lorsque des transactions apportent des modifications à des lignes d'un bloc. Les en-têtes de bloc s'étendent du haut vers le bas.

A l'origine, l'espace libre se situe au milieu du bloc de données. Cela permet l'extension de l'en-tête du bloc et de l'espace des données.
Au début, l'espace libre est contigu, mais les suppressions et les mises à jour peuvent entraîner une fragmentation de l'espace libre dans le bloc. Le serveur Oracle fusionne l'espace libre lorsque cela est nécessaire.

L'espace des données contient des lignes de données. Il peut s'agir de tables de données ou d'index de données. Les données sont insérées dans le bloc du bas vers le haut.

1.3.2. Paramètres d'utilisation de l'espace dans un bloc

Les paramètres de l'espace dans un bloc contrôlent la façon dont l'espace est utilisé dans les segments de données et d'index. Il existe deux types de paramètres :
- les paramètres qui contrôlent la concurrence,
- les paramètres qui contrôlent l'utilisation de l'espace des données.

Les paramètres qui contrôlent la concurrence sont INITRANS et MAXTRANS.
Le paramètre INITRANS indique le nombre initial d'entrées de transaction créées dans un bloc d'index ou un bloc de données.
Les entrées de transaction permettent de stocker des informations sur les transactions qui ont modifié le bloc. Le paramètre INITRANS, dont la valeur par défaut est un (1) pour un segment de données et deux (2) pour un segment d'index, garantit un niveau minimal de concurrence.
Par exemple, si le paramètre INITRANS a la valeur trois (3), cela signifie qu'au moins trois transactions peuvent modifier le bloc simultanément. Des entrées de transaction supplémentaires peuvent être allouées à partir de l'espace libre du bloc pour permettre à davantage de transactions de modifier les lignes du bloc simultanément.

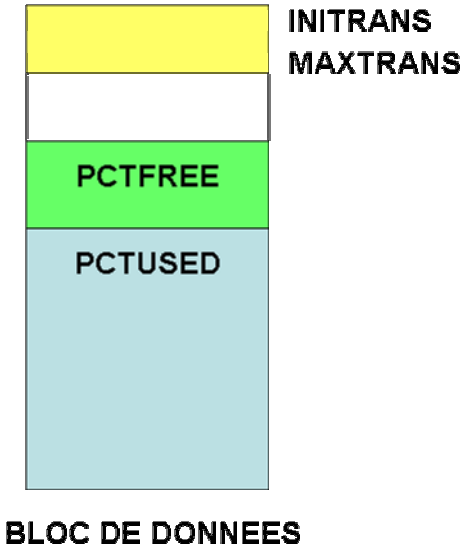
Le paramètre MAXTRANS indique le nombre maximal d'entrées de transaction qui peuvent être créées dans un bloc d'index ou un bloc de données. La valeur par défaut du paramètre MAXTRANS est 255.
Le paramètre MAXTRANS définit le nombre limite de transactions pouvant modifier simultanément un bloc de données ou d'index. La valeur du paramètre MAXTRANS restreint l'utilisation de l'espace pour les entrées de transaction. Cela permet de s'assurer que le bloc dispose de suffisamment d'espace pour les lignes de données ou les index de données.

PCTFREE et PCTUSED sont les paramètres d'utilisation de l'espace dans un bloc qui contrôlent la façon dont l'espace des données est utilisé au sein d'un bloc.

Le paramètre PCTFREE indique, pour chaque bloc de données, le pourcentage d'espace réservé à l'extension due à la mise à jour des lignes dans le bloc. La valeur par défaut de ce paramètre est de 10 pour cent.
Si le paramètre PCTFREE a la valeur 20 pour une instruction CREATE TABLE. Cela signifie que toute insertion de nouvelles données dans ce bloc doit cesser dès lors que l'espace libre est de 20% ou moins. Le reste de l'espace libre peut servir uniquement pour des mises à jour.

Le paramètre PCTUSED indique le pourcentage minimal d'espace utilisé que le serveur Oracle tente de conserver pour chaque bloc de données de la table ; La valeur par défaut de ce paramètre est de 40 pour cent.

Lorsque l'espace occupé par les données dans un bloc de données atteint la limite fixée par la valeur du paramètre PCTFREE, Oracle considère que ce bloc ne peut plus accepter l'insertion de nouvelles lignes. Le bloc reste indisponible pour l'insertion de nouvelles lignes jusqu'à ce que le pourcentage d'espace occupé par les données de ce bloc repasse en dessous de la valeur définie pour le paramètre PCTUSED.
Jusqu'à ce que ce pourcentage repasse en dessous de la valeur définie pour le paramètre PCTUSED, Oracle utilise l'espace libre uniquement pour la mise à jour des lignes contenues dans le bloc de données.
Par exemple, si la valeur du paramètre PCTUSED est égale à 40 pour cent, le bloc peut de nouveau accepter des insertions de lignes dès que son utilisation passe en dessous de 40 pour cent. Les insertions peuvent se poursuivre jusqu'à ce que l'utilisation atteigne 80 pour cent, puis le processus recommence.



1.4.Informations sur la structure de stockage

1.4.1. Recherche d'informations sur les segments

Le nombre courant d'extents et de blocs alloués à un segment peut être obtenu à l'aide de la vue de dictionnaire de données DBA_SEGMENTS.

La vue DBA_SEGMENTS utilise un certain nombre de colonnes.
Elles peuvent être réparties en trois catégories :

- les colonnes qui affichent des informations générales,
- les colonnes qui affichent la taille,
- les colonnes qui affichent les paramètres de stockage.

Les catégories et noms des colonnes sont les suivantes :

Colonnes de la vue DBA_SEGMENTS		
Informations	**Taille**	**Paramètres de stockage**

o OWNER o SEGMENT_NAME o SEGMENT_TYPE o TABLESPACE_NAME	o EXTENTS o BLOCKS	o INTIAL_EXTENT o NEXT_EXTENT o MIN_EXTENT o MAX_EXTENT o PCT_INCREASE

Exemple :

Rechercher les informations permettant de vérifier le nombre courant d'extents et de blocs alloués à un segment. Interroger la vue DBA_SEGMENTS pour extraire les données des colonnes SEGMENT_NAME, TABLESPACE_NAME, EXTENTS et BLOCKS pour les segments appartenant à SCOTT.

```
SQL> SELECT       segment_name, tablespace_name, extents, blocks
  2  FROM   DBA_SEGMENTS
  3  WHERE  owner = 'SCOTT';
SEGMENT_NAME      TABLESPACE_NAME     EXTENTS       BLOCKS
--------------- ---------------------  -----------  -----------
DEPT USER_DATA    1      5
EMP  USER_DATA    1      5
BONUS       USER_DATA    1      5
SALGRADE    USER_DATA    1      5
S_CUSTOMER  USER_DATA    1      5
5 rows selected.
```

1.4.2. Affichage des informations sur les extents utilisés

La vue DBA_EXTENTS permet de vérifier l'espace alloué aux extents dans un segment donné.

La vue DBA_EXTENTS contient un certain nombre de colonnes. Celles-ci peuvent être réparties en deux catégories :

- les colonnes utilisées à des fins d'identification.
- les colonnes affichant l'emplacement et la taille des extents.

Les catégories et noms des colonnes sont décrits ci-dessous :

Colonnes de la vue DBA_EXTENTS	
Informations	**Taille**
o OWNER o SEGMENT_NAME o EXTENT_ID	o TABLESPACE_NAME o RELATIVE_FNO o FILE_ID o BLOCK_ID o BLOCKS

Exemple :

Rechercher les informations relatives aux extents pour le segment EMP appartenant à SCOTT. Interroger la vue DBA_EXTENTS pour extraire les données des colonnes EXTENT_ID, FILE_ID, BLOCK_ID et BLOCKS.

```
SQL> SELECT       extent_id, file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  owner='SCOTT' AND segment_name='EMP';
     EXTENT_ID     FILE_ID       BLOCK_ID      BLOCKS
     ------------ ------------ -----------  -----------
     0      2      12     5
1 row selected.
```

1.4.3. Recherche d'informations sur les extents libres

Pour pouvoir stocker les données, l'administrateur doit connaître la quantité d'espace libre dont il dispose dans les tablespaces.
La vue DBA_FREE_SPACE permet de connaître la quantité d'espace libre dans chaque tablespace de la base de données.
La vue DBA_FREE_SPACE utilise un certain nombre de colonnes pour indiquer l'emplacement des blocs libres dans un tablespace :

- TABLESPACE_NAME
- RELATIVE_FNO
- FILE_ID
- BLOCK_ID
- BLOCKS

Vérifier le nombre d'extents et de blocs libres dans chaque tablespace à l'aide de la vue DBA_FREE_SPACE. Extraire les données des colonnes TABLESPACE_NAME, COUNT(*) et SUM(BLOCKS). Classer ensuite l'affichage en fonction de la colonne TABLESPACE_NAME.

```
SQL> SELECT       tablespace_name, count(*), SUM(blocks)
  2  FROM   dba_free_space
  3  GROUP BY     tablespace_name;
TABLESPACE_NAME   EXTENTS      BLOCKS
----------------------  -----------  -----------
ROLLBACK_DATA           7      2719
SYSTEM            12     2536
TEMPORARY_DATA          1      1023
USER_DATA         1      640
4 rows selected.
```

Cette requête a extrait les informations relatives aux blocs libres pour tous les tablespaces disponibles. Le tablespace ROLLBACK_DATA contient 7 extents libres avec un total de 2719 blocs.

2. Gestion des rollback segments

2.1. Rollback segments : vue d'ensemble

2.1.1. Rollback segment : utilités

Un rollback segment permet de stocker une image avant modification des données.

Une transaction ne peut utiliser qu'un seul rollback segment pour stocker tous les enregistrements de rollback. En revanche, plusieurs transactions peuvent écrire dans un même rollback segment.

L'en-tête du rollback segment contient une table de transactions. Cette table contient une entrée pour chaque transaction qui utilise le rollback segment à un moment donné.

Les rollback segment sont utilisées à différentes fins :

- Pour l'annulation de transactions. Lorsqu'une transaction modifie des données, l'ancienne image de ces données est enregistrée dans un rollback segment. Si la transaction qui a modifié les données est annulée, la valeur d'origine est restaurée. Les informations contenus dans le rollback segment sont utilisées pour reconstruire l' « image avant » des lignes affectées.
- Pour la restauration des transactions. Si une instance Oracle échoue alors que des transactions sont en cours, le serveur Oracle assure la restauration de ces transactions en annulant les modifications non validées après réouverture de la base de données. La restauration des transactions est possible car toutes les modifications apportées aux rollback segments sont également protégées par des fichiers redo log.
- Pour la lecture cohérente d'une instruction donnée. La cohérence de la lecture signifie que seules les modification qui étaient validées au début de l'exécution de l'instruction sont visibles pour les utilisateurs de la base de données.

2.1.2. Rollback segments : types

Le serveur Oracle distingue les rollback segments SYSTEM et non-SYSTEM.

Les modifications apportées aux objets du tablespace SYSTEM sont stockées dans le rollback segment SYSTEM. Ce rollback segment est créé en même temps que la base de données.

L'autre type de rollback segment est le rollback segment non-SYSTEM. L'administrateur de base de données doit créer ce type de rollback segment lorsqu'une base de données contient plusieurs tablespaces.
Une base de données qui contient plusieurs tablespaces nécessite au moins un rollback segment non-SYSTEM. Les « images avant » des modifications apportées aux objets d'un tablespace non-SYSTEM sont stockées dans un rollback segment non-SYSTEM.

Les rollback segments non-SYSTEM peuvent être privés ou publics.

Si les rollback segments privés sont nommés dans le fichier des paramètres à l'aide du paramètre ROLLBACK_SEGMENTS, ils sont automatiquement acquis par une instance Oracle.
Les rollback segments privés peuvent également être acquis manuellement. Pour cela, il suffit de les mettre online à l'aide de la commande `ALTER ROLLBACK SEGMENT`.
Par défaut, tous les rollback segments créés sont privés.

Toute instance peut acquérir un rollback segment public disponible.
Un rollback segment public peut être utilisé avec le serveur parallèle Oracle. Une instance peut acquérir un rollback segment public parmi ceux définis dans la base de données. Toutefois, ce rollback segment ne peut être utilisé que par une instance Oracle à la fois.

Outre les rollback segments privés et publics, il existe des rollback segments en état différé. Ce type de rollback segment est automatiquement créé lorsqu'un utilisateur annule une transaction qui a modifié des données dans un tablespace qui est actuellement offline.
Les rollback segments en état différé contiennent les entrées de rollback qui ne peuvent pas être appliquées à un tablespace parce que celui-ci est offline.
Lorsque le tablespace en question est mis online, les entrées de rollback sont utilisées pour restaurer les anciennes valeurs dans les lignes. Le rollback segment en état différé est supprimé une fois que les anciennes valeurs ont été restaurées.

2.2.Rollback segment : transactions

2.2.1. Rollback segments : utilisation par les transactions

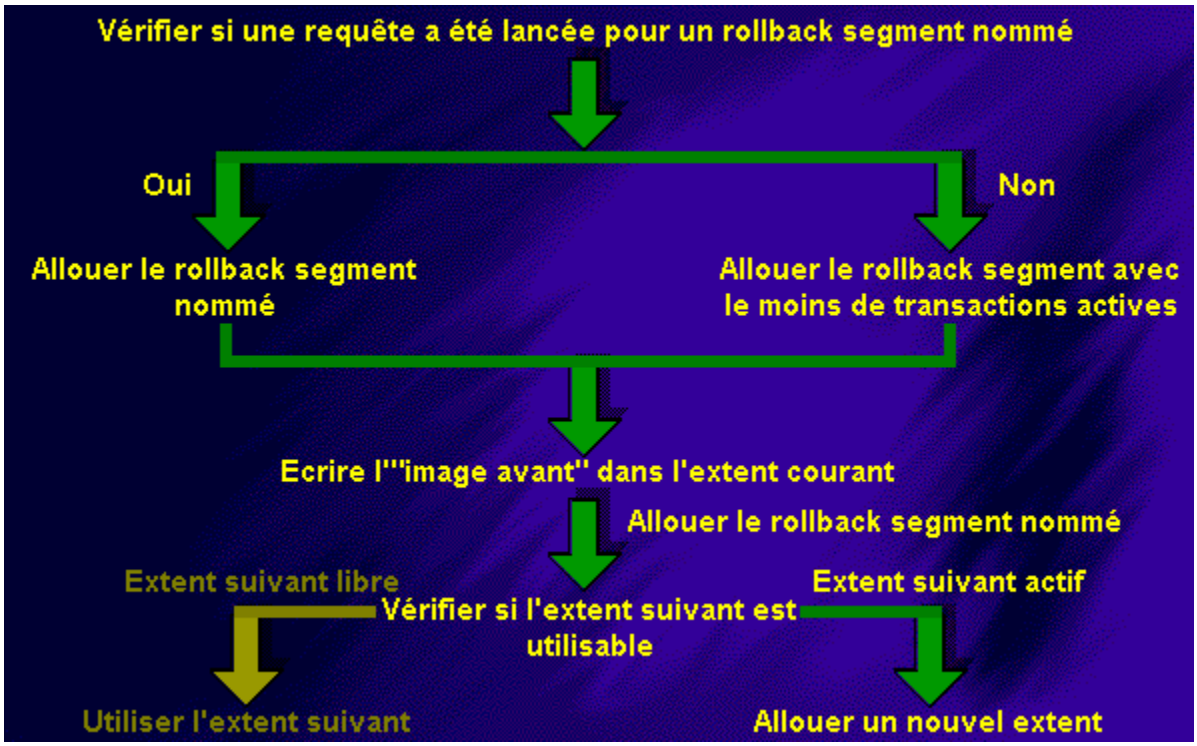
Les translations utilisent les extents d'un rollback segment dans un ordre circulaire, en passant à l'extent suivant quand l'extent courant est plein. Pour une meilleure gestion des rollback segments, il est indispensable de comprendre comment les transactions Oracle utilisent les rollback segments.

Le processus selon lequel les transactions utilisent des rollback segments comprend plusieurs étapes. Ce processus commence au démarrage de la transaction.
Lorsque l'exécution de la transaction démarre, un rollback segment doit lui être affecté. Le serveur Oracle vérifie si une requête a été lancée pour un rollback segment nommé.
L'étape suivante consiste à allouer un rollback segment approprié à la transaction. Si une requête pour un rollback segment nommé a été effectuée, le serveur Oracle alloue ce rollback segment à la transaction.

La commande permettant de demander un rollback segment est :

```
SET TRANSACTION USE ROLLBACK SEGMENT <rollback_segment name> ;
```

Si aucune requête pour un rollback segment nommé n'a été effectuée, le serveur Oracle alloue le rollback segment qui a le moins de transactions actives.
Ensuite, la transaction écrit les informations de rollback, générées par les modifications apportées aux données, dans l'extent courant du rollback segment. L'extent courant est celui dans lequel d'autres transactions sont également en train d'écrire. Un pointeur est utilisé pour indiquer l'emplacement où les entrées seront écrites.
Lorsqu'un extent est plein, le pointeur doit passer à l'extent disponible suivant. A ce stade, le serveur Oracle recherche l'extent utilisable suivant dans le rollback segment.
Si l'extent suivant est libre ou inactif, il peut être utilisé. Un extent est libre ou inactif s'il n'est utilisé par aucune transaction active. Si l'extent suivant est utilisé, le pointeur se place dans cet extent et les transactions commencent à écrire dedans.
Si l'extent suivant est actif, la transaction ne peut pas utiliser cet extent. En outre, le pointeur ne peut pas ignorer un extent et passer à un autre extent inactif. Le serveur Oracle alloue un extent supplémentaire pour le rollback segment. La transaction écrit alors dans cet extent supplémentaire.



L'allocation d'un extent supplémentaire est appelée une extension. Un rollback segment peut s'étendre de cette façon jusqu'à ce qu'il atteigne le nombre d'extents spécifié dans son paramètre de stockage MAXEXTENTS ou jusqu'à ce que tout l'espace du tablespace soit utilisé.

2.2.2. Rollback segment : rétrécissement

Le serveur Oracle peut automatiquement libérer l'espace inutilisé dans un rollback segment.
Le serveur Oracle peut rétrécir un rollback segment jusqu'à ce que sa taille atteigne la valeur spécifiée dans le paramètre de stockage OPTIMAL du rollback segment.
Cette taille optimale doit être définie en fonction de l'espace nécessaire à une transaction moyenne et du nombre estimé de transactions simultanées par rollback segment.

Le rétrécissement d'un rollback segment se produit lorsque le pointeur d'un rollback segment passe à l'extent suivant. Les extents sont libérés jusqu'à ce qu'un extent actif soit trouvé ou que le rollback segment ait atteint sa taille optimale.
Les rollback segments rétrécissent dans certaines situations :

- lorsque sa taille courante excède la valeur spécifiée pour le paramètre OPTIMAL.
- si un rollback segment contient des extents inactifs contigus.

Par exemple, pour mieux comprendre la notion de rétrécissement des rollback segments. Le pointeur d'un rollback segment passe de l'extent 1 à l'extent 2 si l'extent 2 est inactif. Ensuite, il vérifie l'extent 3. Si l'extent 3 est inactif et que la taille du rollback segment excède la taille optimale, l'extent 2 est libéré. Une fois l'extent 2 libéré, le pointeur passe à l'extent suivant. Le processus de rétrécissement continue jusqu'à ce que la taille du rollback segment atteigne la valeur optimale ou que l'extent suivant soit actif.

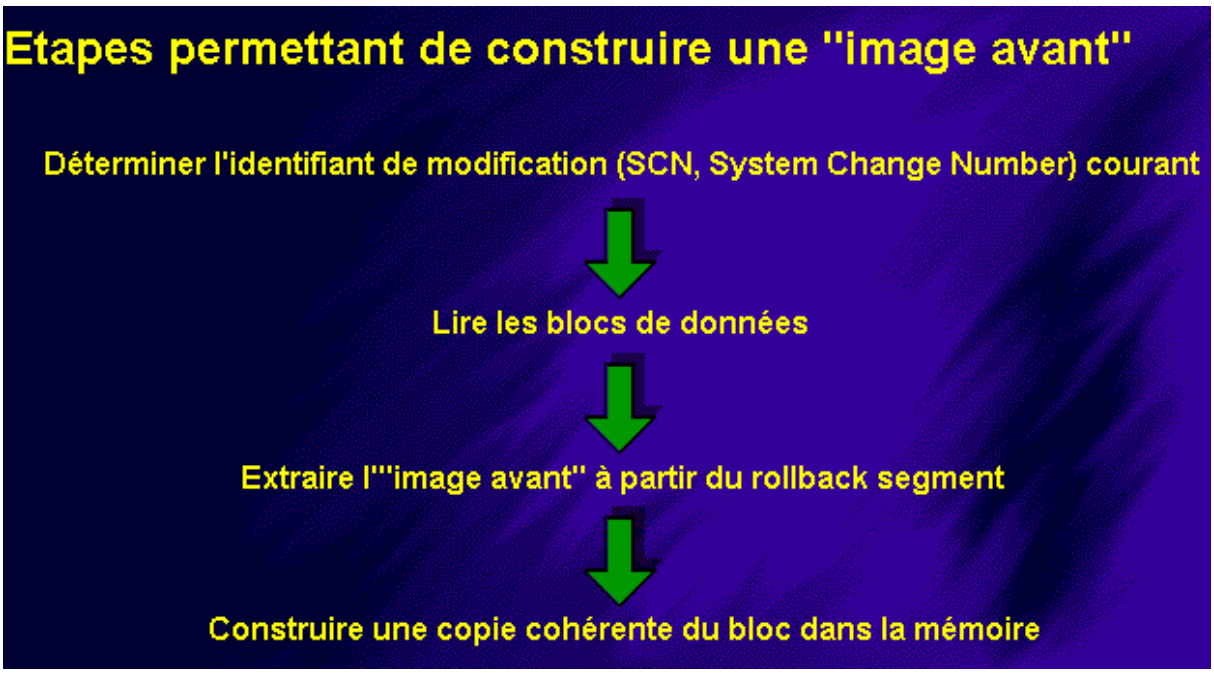
Le serveur Oracle libère les extents inactifs les plus anciens dans la mesure où ces extents ont peu de chance d'être utilisés pour la cohérence de la lecture.

2.2.3. Rollback segment : cohérence de la lecture

Le serveur Oracle garantit la cohérence d'un ensemble de données lus par une instruction par rapport à un moment donné.

Pour construire une « image avant » des données, le serveur Oracle utilise les informations contenues dans un rollback segment et les informations courantes de la table.
Une fois l'exécution de l'instruction terminée et la transaction validée, les modifications apportées par la transaction deviennent permanentes. Les instructions dont l'exécution commence après la validation d'une transaction peuvent lire les modifications apportées par cette transaction.

Cet exemple illustre le processus permettant de garantir une lecture cohérente :
Une requête longue est exécutée alors que plusieurs modifications sont en train d'être apportées. Si un bloc reçoit des modifications qui n'étaient pas encore validées au début de l'exécution de la requête, le serveur Oracle construit une « image avant » des lignes modifiées dans le bloc.
L' « image avant » du bloc est construite en plusieurs étapes.
Tout d'abord, lorsque le serveur Oracle démarre l'exécution d'une instruction SELECT, il détermine l'identifiant de modification (SCN, System Change Number) courant.
Ensuite, le serveur Oracle lit les blocs de données nécessaires. Il recherche les lignes qui n'étaient pas validées au début de l'exécution de l'instruction.
Si les blocs de données contiennent des modifications qui n'étaient pas encore validées au début de l'exécution de la transaction, le serveur Oracle extrait l' « image avant » de ces modifications non validées à partir du rollback segment.
Enfin, le serveur Oracle construit une copie cohérente du bloc en appliquant l' « image avant » stockée dans le rollback segment aux lignes affectées dans le bloc.



Au début d'une transaction, l'administrateur utilise la commande `SET TRANSACTION READ ONLY` pour demander la cohérence de la lecture. Lorsque cette commande est exécutée, le serveur Oracle tente de fournir une « vue avant » (cohérente en lecture) des données à l'aide des rollback segments, et ce jusqu'à la fin de la transaction.

2.3.Configuration des rollback segments

2.3.1. Planification des rollback segments

L'administrateur doit planifier le nombre de rollback segments à créer dans la base de données, ainsi que la taille. Un certain nombre de facteurs influent cette planification :

- environnement de l'application
- type de transaction
- volume de données traitées

Les environnements transactionnels, par exemple OLTP (Online Transaction Processing) ou batch, ont une incidence sur le nombre, et la taille, des rollback segments nécessaires dans une base de données. Chaque rollback segment comporte un en-tête. Cet en-tête contient les entrées de table de transactions définissant l'état de chaque transaction qui écrit dans le rollback segment.
Chaque transaction qui utilise un rollback segment met à jour de transactions de façon régulière. Cela peut provoquer une contention sur l'en-tête, surtout dans un environnement OLTP.

Un environnement OLTP se caractérise par des transactions courtes. Par conséquent, dans cet environnement, il est préférable d'avoir de nombreux rollback segments de petite taille. Si possible, il faut créer un rollback segment pour chaque groupe de quatre transactions simultanées.

Un environnement batch se caractérise par des transactions volumineuses. Ces transactions nécessitent des rollback segments de grande taille. Par conséquent, dans cet environnement, il est préférable de créer des rollback segments de grande taille.

Les autres facteurs qui influencent la planification des rollback segments sont le type de transaction effectuée et la quantité de données traitées. Ces facteurs ont une incidence sur la taille des rollback segments.

La taille d'un rollback segment peut correspondre au nombre maximal d'octets d'informations de rollback estimé pour chaque transaction, multiplié par le nombre estimé de transactions simultanées qui utiliseront ce rollback segment.

La taille du bloc de rollback généré au cours d'une transaction dépend du type de transaction, notamment s'il s'agit d'une insertion ou d'une suppression. Par exemple, l'insertion d'un enregistrement dans une table génère moins d'informations de rollback que la suppression du même enregistrement.

Cette différence de taille au fait qu'une transaction nécessitant l'insertion de données stocke uniquement le ROWID dans le bloc de rollback. En revanche, une transaction nécessitant la suppression stocke toutes les données supprimées.

L'autre facteur à prendre en compte est le volume de données traitées. Une transaction qui traite 100 enregistrements va probablement générer davantage d'informations de rollback qu'une transaction qui traite 10 enregistrements.

Avant de définir la taille de vos rollback segments, il faut prendre en compte les règles suivantes :

- Pour évaluer la taille d'un rollback segment, exécutez la transaction la plus volumineuse que vous prévoyez et vérifiez la taille du rollback segment.
- Pou limiter l'extension dynamique des rollback segments, créez des rollback segments contenant de nombreux extents. La valeur recommandée pour le paramètre MINEXTENTS est de 20 extents.

2.3.2. Création de rollback segments

Une base de données comportant plusieurs tablespaces nécessite au moins un rollback segment en plus du rollback segment SYSTEM créé par défaut.

La syntaxe de la commande SQL permettant de créer un rollback segment est la suivante :

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
[TABLESPACE tablespace]
[storage_clause];
```

Exemple :

Créer un rollback segment RB dans le tablespace RBS. Les paramètres INITIAL, NEXT, OPTIMAL, MINEXTENTS et MAXEXTENTS ont respectivement la valeur 100K, 100K, 2M, 20 et 100.

```
SQL> CREATE ROLLBACK SEGMENT rb
  2  TABLESPACE rbs1
  3  STORAGE ( INITIAL 100K NEXT 100K OPTIMAL 2M
  4  MINEXTENTS 20 MAXEXTENTS 100 ) ;
segment rollback created.
```

Lors de la création d'un rollback segment, un certain nombre de restrictions doivent être prises en compte. Ces restrictions sont les suivantes :

- La nature d'un rollback segment, privé ou public, ne peut plus être modifié par la suite.
- Le paramètre MINEXTENTS ne peut pas avoir une valeur inférieure à 2 pour un rollback segment.
- Le paramètre PCTINCREASE ne peut pas être spécifié pour un rollback segment et sa valeur est toujours 0.
- La valeur OPTIMAL, si elle est spécifiée, doit au moins être égale à taille initiale du rollback segment

Exemple :

Créer un rollback RB_01 dans le tablespace RBS. Spécifier les paramètres de stockage INITIAL et NEXT pour ce rollback segment.

```
SQL> CREATE ROLLBACK SEGMENT RB_01
  2  TABLESPACE rbs STORAGE ( INITIAL 100K NEXT 100K);
rollback segment created.
```

Il est également possible de créer un rollback segment qui utilise les mêmes tablespaces et paramètres qu'un rollback segment existant à l'aide d'Oracle Storage Manager.

Lors de la définition des paramètres appropriés pour les rollback segments, il faut prendre en compte les règles suivantes :

- Utilisez la règle INITIAL = NEXT pour les rollback segments afin de vous assurer que tous les extents sont de la même taille
- Définissez la valeur du paramètre OPTIMAL en fonction de l'espace nécessaire pour une transaction moyenne. Si vous ne disposez pas des informations nécessaires, indiquez une valeur correspondant à la taille initiale ; vous la modifierez ultérieurement.
- Evitez d'attribuer la valeur UNLIMITED au paramètre MAXEXTENTS. En effet, cela pourrait provoquer une extension inutile d'un rollback segment et même de fichiers de données à la suite d'une erreur d'un programme.

- Placez vos rollback segments dans un tablespace distinct qui leur est réservé, afin de limiter la contention et la fragmentation.

La création de plusieurs rollback segments peut limiter la contention pour les rollback segments entre les transactions.

2.3.3. Mise en ligne d'un rollback segment

Lors de sa création, le rollback segment est offline et ne peut pas être utilisé pour stocker des données de rollback. Pour qu'un rollback segment puisse être utilisé par des transactions, il doit être mis online.

La syntaxe de la commande SQL permettant de mettre un rollback segment online et de le rendre disponible est la suivante :

```
ALTER ROLLBACK SEGMENT rollback_segment ONLINE;
```

Exemple :

Le rollback segment RB_01 est offline par défaut. Le mettre online pour qu'il puisse être utilisé par des transactions.

```
SQL> ALTER ROLLBACK SEGMENT "RB_01" ONLINE;
Rollback segment altered.
```

Le nombre de rollback segments pouvant être mis online par une instance Oracle est limité par la valeur du paramètre MAX_ROLLBACK_SEGMENTS définie dans le fichier de paramètres. Cette valeur doit être supérieure d'une unité au nombre de rollback segments non-SYSTEM nécessaires pour l'instance Oracle.

Lorsqu'une instance Oracle est arrêtée puis redémarrée, l'état d'un rollback segment peut être offline ou online.

Pour s'assurer qu'un rollback segment donné sera bien toujours mis online par une instance Oracle, il faut spécifier le nom de ce rollback segment dans le fichier des paramètres. Le format utilisé est le suivant :

```
ROLLBACK_SEGMENTS = (tablespace_names)
```

Exemple :

```
ROLLBACK_SEGMENTS = (RB_01)
```

2.3.4. Acquisition d'un rollback segment

Une instance Oracle acquiert des rollback segments lorsqu'elle ouvre une base de données. L'administrateur doit s'assurer que l'instance Oracle dispose de suffisamment de rollback segments lors de l'ouverture de la base de données.

Le processus permettant à une instance Oracle d'acquérir des rollback segments lorsqu'elle ouvre une base de données comprend plusieurs étapes :

- L'instance Oracle acquiert tous les segments privés qui sont nommés dans le paramètre d'initialisation ROLLBACK_SEGMENTS
- Calculer le nombre de rollback segments nécessaires

  Nombre de rollback segments nécessaires = valeur du paramètre TRANSACTIONS / Valeur du paramètre TRANSACTIONS_PER_ROLLBACK_SEGMENT

Ces paramètres sont simplement utilisés par l'instance Oracle pour estimer le nombre de rollback segments nécessaires. Ils ne limitent en aucun cas le nombre de transactions dans l'instance Oracle.

- L'instance Oracle vérifie si le nombre de rollback segments non-SYSTEM acquis est au moins égal au nombre de rollback segments estimé.
- La quatrième étape implique une décision.
  Si le nombre de rollback segments nécessaires est supérieur au nombre de rollback segments non-SYSTEM disponibles pour l'instance Oracle, le serveur Oracle acquiert des rollback segments publics supplémentaires.
  Si le nombre de rollback segments nécessaires est inférieur ou égal au nombre de rollback segments non-SYSTEM disponibles pour l'instance Oracle, l'instance Oracle n'a pas besoin d'acquérir des rollback segments supplémentaires.

2.4.Gestion des rollback segments

2.4.1. Modification des paramètres de stockage

Les paramètres de stockage définis pour un rollback segment peuvent être modifiés. Cette opération peut être utile si le nombre d'extents nécessaires est supérieur à celui initialement prévu ou si l'estimation des besoins moyens doit être modifiée.

La syntaxe de la commande SQL permettant de modifier les paramètres de stockage d'un rollback segment est la suivante :

```
ALTER ROLLBACK SEGMENT rollback_segment storage_clause;
```

La commande ALTER ROLLBACK SEGMENT est normalement utilisée pour redéfinir les paramètres MAXEXTENTS et OPTIMAL d'un rollback segment. Le paramètre INITIAL d'un rollback segment ne peut pas être redéfini.

Exemple :

Les transactions dans la base de données ont généré plus d'informations de rollback que prévu. Pour éviter de fréquentes allocations et libérations d'extents, il faut augmenter la valeur du paramètre OPTIMAL pour le rollback segment RB_01.

```
SQL> ALTER ROLLBACK SEGMENT "RB_01"
  2  STORAGE ( OPTIMAL 500K ) ;
rollback segment altered.
```

2.4.2. Rétrécissement des rollback segments

Il est possible de garantir une utilisation optimale de l'espace dans un rollback segment en libérant manuellement l'espace inutilisé. Pour cela, il est possible de réduire la taille du rollback segment en lui attribuant la valeur de son paramètre OPTIMAL. Sinon, il est également possible de spécifier la taille souhaitée pour votre rollback segment rétréci.

La syntaxe de la commande SQL permettant de libérer l'espace inutilisé dans un rollback segment est la suivante :

```
ALTER ROLLBACK SEGMENT rollback_segment SHRINK [TO integer
[K|M]];
```

Lorsqu'un nombre entier (*integer*) est spécifié dans cette commande, le serveur Oracle essaie de libérer de l'espace inutilisé afin de réduire la taille du rollback segment à la valeur indiquée.

Si aucun nombre n'a été spécifié, le serveur Oracle libère l'espace inutilisé qui excède la taille optimale. Pour cela, le paramètre OPTIMAL doit avoir été préalablement spécifié pour le rollback segment.

Exemple :

Le paramètre OPTIMAL du rollback segment RB_01 est défini à l'aide de la clause de stockage. Pour limiter le gaspillage d'espace dans le rollback segment RB_01, il faut libérer de l'espace inutilisé jusqu'à ce que la taille de ce segment soit réduite à la valeur du paramètre OPTIMAL.

```
SQL> ALTER ROLLBACK SEGMENT rb_01 SHRINK;
Rollback segment altered.
```

Toutefois, il existe une restriction quant au rétrécissement des rollback segments. Le rétrécissement s'arrête si un extent ne peut pas être libéré parce qu'il est actif.

2.4.3. Mise offline d'un rollback segment

L'administrateur souhaite supprimer un rollback segment parce qu'il est devenu inutile. Avant de supprimer un rollback segment, il faut obligatoirement le mettre offline.
La syntaxe de la commande SQL permettant de mettre un rollback segment offline est la suivante :

```
ALTER ROLLBACK SEGMENT rollback_segment OFFLINE;
```

Lors de l'exécution de l'instruction ALTER ROLLBACK SEGMENT, si des transactions utilisent le rollback segment concerné, l'état de ce rollback segment passe à PENDING OFFLINE. Cet état peut être constaté à l'aide de la vue dynamique sur les performances V$ROLLSTAT. Une fois que toutes les transactions sont terminées, le rollback segment est mis offline.

Exemple :

Le rollback segment RB_01 doit être supprimé de la base de données. Avant de supprimer ce rollback segment, il faut le mettre offline.

```
SQL> ALTER ROLLBACK SEGMENT "RB_01" OFFLINE;
Rollback segment altered.
```

2.4.4. Suppression d'un rollback segment

La taille spécifiée dans le paramètre INITIAL d'un rollback segment doit être modifiée car la taille moyenne des transactions a augmenté. Pour cela, il faut recréer le rollback segment en commençant par le supprimer.

La syntaxe de la commande SQL permettant de supprimer un rollback segment est la suivante :

```
DROP ROLLBACK SEGMENT rollback_segment;
```

Exemple :

Supprimer le rollback segment RB_01 après l'avoir passé offline.

```
SQL> ALTER ROLLBACK SEGMENT "RB_01" OFFLINE;
Rollback segment altered.
SQL> DROP ROLLBACK SEGMENT "RB_01";
Rollback segment dropped.
```

Après avoir supprimé le rollback segment, il peut être recréé avec de nouveaux paramètres INITIAL et NEXT.
Un rollback segment peut être également supprimé lorsqu'il n'est plus utile.

2.5.Informations générales sur rollback segments

2.5.1. Informations générales sur les rollback segments

Les performances peuvent diminuer sir le nombre de rollback segments disponibles pour les transactions est insuffisant. Dans ce cas, il faut commencer par vérifier s'il reste des rollback segments offline avant d'en créer de nouveaux. Les rollback segments offline peuvent être mis online et ainsi rendus disponibles pour les transactions.

Les informations concernant les rollback segments offline ne peuvent être visualisées que dans la vue DBA_ROLLBACK_SEGS. Il n'est pas possible d'extraire ces informations des vues dynamiques sur les performances. En effet, ces vues affichent uniquement les rollback segments qui sont actuellement utilisés par une instance Oracle.
La vue DBA_ROLLBACK_SEGS affiche des informations générales sur les rollback segments. Ces informations comprennent l'identification, l'emplacement, le type et l'état des rollback segments.
Un rollback segment peut être identifié par ses paramètres SEGMENT_ID et SEGMENT_NAME.

L'emplacement d'un rollback segment est spécifié par le nom du tablespace dans lequel il a été créé.

Un rollback segment peut être de type PUBLIC ou SYS. Le type SYS correspond à un rollback segment privé et le type PUBLIC à un rollback segment public.

L'état d'un rollback segment peut être ONLINE ou OFFLINE.

2.5.2. Statistiques sur les rollback segments

L'administrateur se doit de surveiller les différents objets de la base de données. Pour optimiser les performances, il peut contrôler la taille actuelle des rollback segments par rapport à leur taille optimale. Pour cela, il a la possibilité d'interroger les vues V$ROLLNAME et V$ROLLSTAT.

Les noms des colonnes de la vue dynamique sur les performances V$ROLLNAME et les données qu'elles contiennent sont les suivantes :

- USN : numéro de rollback segment
- NAME : nom du rollback segment

Les noms des colonnes de la vue dynamique sur les performances V$ROLLSTAT et les données qu'elles contiennent sont les suivantes :

- USN : numéro du rollback segment
- EXTENTS : nombre d'extents dans le rollback segment
- RSSIZE : taille actuelle du segment, en octets
- XACTS : nombre de transactions utilisant le rollback segment
- OPTSIZE : valeur OPTIMAL pour le rollback segment
- HWMSIZE : High water mark (taille maximale atteinte par le segment depuis le démarrage)
- STATUS : état du rollback segment

Une jointure des vues V$ROLLSTAT et V$ROLLNAME peut être faite pour afficher des informations statistiques sur les rollback segments actuellement utilisés par une instance Oracle.

Exemple :

> Interroger les vues V$ROLLNAME et V$ROLLSTAT pour extraire les colonnes NAME, EXTENTS, RSSIZE et OPTSIZE.

```
SQL> SELECT       n.name, s.extents,s.tssize, s.optsize
  2  FROM   V$ROLLNAME, V$ROLLSTAT s
  3  WHERE  n.usn = s.usn;
NAME        EXTENTS      RESIZE OPTSIZE
----------------------   ----------   ----------   ----------
SYSTEM            8      407552
RB_01             2      28672  26624
RB_05             2      18432  26624
3 rows selected.
```

2.5.3. Activité courante des rollback segments

Lors de l'exécution de certaines opérations de maintenance, l'administrateur a besoin d'identifier les sessions qui utilisent actuellement des rollback segments. Ces informations peuvent être extraites à partir des vues V$SESSION et V$TRANSACTION.

Voici certaines colonnes de la vue V$SESSION :

- SADDR : adresse de la session
- USERNAME : nom de l'utilisateur
- SID : identification de la session
- SERIAL# : numéro de série

Voici certaines colonnes de la vue V$TRANSACTION :

- SES_ADDR : adresse de la session
- XIDUSN : numéro de rollback segment utilisé par la transaction
- UBAFIL, UBABLK, UBASQN, UBAREC : emplacement dans lequel la transaction écrit actuellement
- USED_UBLK : nombre de blocs de rollback générés
- START_UEXT : extent du rollback segment à partir duquel la transaction a commencé à écrire
- START_UBAFIL : numéro du fichier de rollback segment à partir duquel la transaction a écrit

Exemple :

Rechercher les informations concernant le nom de l'utilisateur qui exécute la transaction, le numéro du rollback segment utilisé par la transaction et le nombre de blocs de rollback générés par cette transaction.

```
SQL> SELECT s.username, t.xidusn,t.used_ublk
  2  FROM   V$SESSION s, V$TRANSATION t
  3  WHERE  s.saddr = t.ses_addr;
USERNAME          XIDUSN USED_UBLK
---------------------------   ---------    -----------
SCOTT             5      260
1 row selected.
```

2.6.Dépannage des rollback segments

2.6.1. Espace insuffisant pour les transactions

Une transaction ne peut écrire que dans un seul rollback segment et risque donc d'échouer si ce rollback segment ne dispose pas de l'espace nécessaire. Cette erreur peut se produire pour diverses raisons.
Si l'espace disponible dans un rollback segment est insuffisant pour une transaction, l'erreur ORA-1562 est générée. Ce problème peut avoir diverses causes :

- L'espace dans le tablespace peut être insuffisant. Ce type de problème génère une erreur ORA-1560. Pour résoudre ce problème, il faut augmenter l'espace disponible dans ce tablespace :

  - Etendre les fichiers de données existants ou
  - Ajouter de nouveaux dans le tablespace ou
  - Autoriser une opération AUTOEXTEND pour ces fichiers de données.
- La limite MAXEXTENTS a été atteinte. Dans ce cas, aucun extent supplémentaire ne peut être alloué au rollback segment concerné. L'erreur ORA-1628 est générée si le manque d'espace disponible dans un rollback segment pour une transaction provient de ce type de problème. Pour résoudre ce problème, il faut :
  - Augmenter la valeur de MAXEXTENT pour le rollback segment concerné ou
  - Recréer des rollback segments avec des extents plus grands. Pour récréer un rollback segment, il faut d'abord le supprimer.

2.6.2. Erreurs liées à la cohérence de la lecture

Le serveur Oracle garantit qu'une instruction traite uniquement les données qui étaient validées au début de l'exécution de cette instruction. Les modifications validées après le début de l'instruction ne seront pas prises en compte par cette instruction.

Si le serveur Oracle ne peut pas construire une "image avant" des données, l'erreur ORA-1555 SNAPSHIT TOO OLD est générée.
L'administrateur doit être capable d'analyser les causes d'une erreur liée à lé cohérence de la lecture et de trouver la solution adéquate.
Une erreur liée à la cohérence de la lecture peut se produire si la transaction qui a apporté les modifications a été validée et que certains événements provoquant des erreurs sont survenus.

Deux types de problèmes peuvent provoquer une erreur de cohérence. Soit l'entrée de transaction dans l'entête du rollback segment a été réutilisée, soit l'"image avant" dans le rollback segment a été annulée par une autre transaction.
Pour limiter les problèmes de cohérence de lecture, il faut :

- Créer des rollback segments en définissant un paramètre MINEXTENTS plus élevé ou
- Créer des rollback segments avec des extents plus grands ou
- Créer des rollback segments en définissant un paramètre OPTIMAL plus élevé.

Toutefois, les erreurs de cohérence de lecture ne peuvent pas être évitées en augmentant la valeur du paramètre MAXEXTENTS.

2.6.3. Session bloquante

Unes session bloquante peut entraîner une extension non maîtrisée d'un rollback segment.
L'administrateur doit être capable de trouver une solution appropriée au problème des sessions bloquantes.
Lorsqu'une transaction écrit dans un extent d'un rollback segment et que cet extent est plein, le serveur Oracle tente de réutiliser l'extent suivant du rollback segment.
Si l'extent suivant contient ne serait-ce qu'une seule entrée de rollback générée après une transaction active, il ne peut être utilisé.
En outre, le pointeur du rollback segment ne peut pas ignorer un extent et continuer à écrire la transaction dans un autre extent inactif. Par conséquent, des extents supplémentaires sont alloués au rollback segment.
Il est possible que la transaction bloquante soit inactive depuis longtemps. Dans ce cas, l'administrateur de base de données doit intervenir pour terminer la transaction, de préférence en rappelant à l'utilisateur qu'il doit mettre fin à cette transaction. Il peut également tuer la transaction inactive à l'origine de la session bloquante dans le rollback segment.

Pour rechercher les éventuels transactions bloquantes, l'administrateur peut utiliser les vues V$ROLLSTAT, V$SESSION et V$TRANSACTION.

Exemple:

```
  SQL> SELECT s.sid, s.serial#,t.start_time,t.xidusn,s.username
    2  FROM   V$SESSION s, V$TRANSATION t, V$ROLLSTAT r
    3  WHERE  s.saddr = t.ses_addr
    4  AND    t.xidusn = r.usn
    5  AND    ((r.curext = t.start_uext-1)
    6  OR     ((r.curext = r.extents-1)
    7  AND    t.start_uext = 0));
       SID    SERIAL START_TIME    XIDUSN USERNAME
       --------     ----------   ------------------ --------     ----
------------
       9      18     04/21/98 15:17:18   4      SCOTT
       14     8      04/21/98 15:18:09   4      SCOTT
```

2.6.4. Erreur lors de la mise offline d'un tablespace

Au cours d'une opération de maintenance sur une base de données, l'administrateur essaie de mettre un tablespace offline. Toutefois, si ce tablespace contient un ou plusieurs rollback segments actifs, il ne peut pas être mis offline. La session qui tente d'exécuter l'instruction recevra alors une erreur. Cette erreur peut être résolu en effectuant une série d'étapes :

- exécuter une requête pour identifier les rollback segments online dans ce tablespace. ces informations peuvent être extraites de la vue DBA_ROLLBACK_SEGS
- mettre offline tous les rollback segments online du tablespace à l'aide de la commande :

  ALTER ROLLBACK SEGMENT *rollback_segment_name* **OFFLINE;**

- Localiser les transactions qui utilisent actuellement ces rollback segments online. Ces informations sont obtenues en recherchant dans la vue V$ROLLSTAT les rollback segments dont l'état est PENDING OFFLINE.
- Identifier les sessions qui utilisent un rollback segment dont l'état est PENDING OFFLIE en effectuant une jointure des vues V$SESSION et V$TRANSACTION.
- Mettre fin, si c'est nécessaire, aux sessions qui ont des transactions actives. Cette opération peut être effectuée à l'aide de la commande ALTER SYSTEM.
- Mettre le tablespace offline.

3. Gestion des segments temporaires

3.1. Concepts liés aux segments temporaires

3.1.1. Types de segments temporaires

Lors du traitement de requêtes, le serveur Oracle nécessite souvent un espace de travail temporaire pour les étapes intermédiaires de l'exécution d'une instruction SQL. Parmi ces étapes intermédiaires figure le tri des données. Pour le tri, le serveur Oracle peut allouer automatiquement de l'espace disque sous forme de segment temporaire.
Un segment temporaire est créé et utilisé par le serveur Oracle dans le tablespace alloué à l'utilisateur qui effectue l'opération. Toutefois, si le tri peut être effectué dans la mémoire, aucun segment temporaire n'est créé.
La quantité de mémoire utilisée par un processus pour une opération de tri est déterminée par la valeur du paramètre SORT_AREA_SIZE. Si le volume des données à trier excède la valeur de ce paramètre, les données sont triées en plusieurs fois. Les résultats intermédiaires de chaque opération de tri sont stockés sur le disque.
Voici des exemples d'instructions pouvant nécessiter des segments temporaires pour des opérations de tri :

- SELECT … ORDER BY
- CREATE INDEX
- SELECT DISTINCT
- SELECT … UNION
- SELECT … GROUP BY

Les segments temporaires peuvent être créés dans différents tablespaces en fonction de l'utilisateur à l'origine du tri. Ils sont créés dans un tablespace permanent ou temporaire.
Le type de tablespace dans lequel un segment temporaire est créé détermine ses caractéristiques.

Segment temporaire dans un tablespace permanent :
L'allocation et la libération de segments temporaires sont des opérations fréquentes. Par conséquent, les tablespaces devant contenir des segments temporaires devraient être du type TEMPORARY.
Si un tablespace permanent est utilisé pour les opérations de tri, une instance Oracle peut avoir plusieurs segments temporaires dans ce tablespace.
Des segments temporaires sont créés dans un tablespace permanent uniquement quand les deux conditions suivantes sont remplies :

- Lorsqu'une instruction nécessitant de l'espace sur le disque pour un tri est exécutée par un utilisateur.
- Lorsqu'un tablespace permanent a été affecté à l'utilisateur qui exécute l'instruction pour les opérations de tri.

L'espace disque est nécessaire pour le tri parce que l'espace mémoire défini par SORT_AREA_SIEZ est trop faible.

Les segments temporaires créés dans des tablespaces permanents ont des caractéristiques bien précises. Tout d'abord, des segments temporaires sont créés pour chaque transaction dès que nécessaire. Ensuite, des segments temporaires sont demandés par le processus System Monitor (SMON) à la fin d'une instruction SQL. L'espace libéré par le segment temporaire peut être utilisé par d'autres objets.
Quand un tablespace permanent est utilisé pour une opération de tri, l'espace libre dans ce tablespace peut être fragmenté de façon importante. Par conséquent, il est conseillé d'utiliser le tablespace en question uniquement pour les opérations de tri.

Segment temporaire dans un tablespace temporaire :

Le serveur Oracle peut utiliser des segments temporaires dans un tablespace temporaire. Ces segments temporaires sont également appelés segments de tri.
Une instance Oracle crée un seul segment temporaire pour chaque tablespace temporaire. Un nombre illimité d'extents peut être alloué à un segment temporaire créé dans un tablespace temporaire.

Un segment temporaire est créé dans un tablespace temporaire lorsque le premier tri sur disque est effectué dans une instance Oracle après le démarrage.
Les informations concernant un segment temporaire sont gérées par le serveur Oracle dans une zone système globale appelée Sort Extent Pool. En fonction des informations contenues dans cette zone, plusieurs transactions nécessitant de l'espace pour des opérations de tri peuvent réutiliser les extents de ce segment temporaire.
Un segment temporaire d'un tablespace temporaire est libéré lors de l'arrêt d'une instance Oracle. Cela permet de réduire le nombre d'opérations d'allocation et de libération d'extents générées par les instructions qui nécessitent des tris. Ainsi les performances des tris qui ne peuvent pas être effectués dans la mémoire peuvent être améliorées.

3.1.2. Règles concernant les segments temporaires

Certaines règes aident à définir les paramètres des tablespaces contenant des segments temporaires.

- Définir des tablespaces temporaires différents en fonction des besoins de tri.
  → Cette utilisation améliore la concurrence et diminue le nombre d'allocations et de libérations d'espace.
- Spécifier les valeur de stockage par défaut INITIAL = NEXT = multiple de SORT_AREA_SIZE + 1 bloc Oracle d'entête et PCTINCREASE = 0.
  Spécifier le paramètre MAXEXTENTS pour les segments temporaires créés dans des tablespaces permanents. Le paramètre MAXENTENTS n'affecte pas les segments temporaires créés dans des tablespaces temporaires.

Il st conseillé de créer des tablespaces avec des clauses de stockage par défaut différentes et de les affecter aux utilisateurs en fonction de leurs besoins en opérations de tri. Cela permet d'utiliser l'espace disponible dans la base de données de façon optimale.

3.2.Informations sur les segments temporaires

3.2.1. Statistiques sur les segments temporaires

Pour optimiser les performances, l'administrateur peut surveiller l'utilisation des segments temporaires en recherchant et en analysant les statistiques les concernant.
Les vues utilisées pour extraire les statistiques sur les segments temporaires sont DBA_SEGMENTS, V$SORT_USAGE et V$SORT_SEGMENT.

La vue DBA_SEGMENTS permet d'extraire des informations sur les deux types de segments temporaires créés dans des tablespaces temporaires sont également appelés segments de tri.

La vue V$SORT_USAGE permet d'extraire des informations sur les segments temporaires qui ont actuellement des tris actifs dans une instance Oracle.

La vue V$SORT_SEGMENT permet d'extraire l'état de la zone Sort Extent Pool utilisée par une instance Oracle.

Voici la liste partielle des colonnes contenues dans la vue V$SORT_SEGMENT :

- TABLESPACE_NAME
- EXTENT_SIZE
- TOTAL_EXTENTS
- TOTAL_BLOCKS

- USED_EXTENTS
- USED_BLOCKS
- FREE_EXTENTS
- FREE_BLOCKS
- MAX_SORT_SIZE
- MAX_SORT_BLOCKS

Pour estimer la taille des tablespaces temporaires nécessaires au stockage des segments de tri, il faut connaître le nombre d'extents et de blocs utilisés par l'opération de tri la plus volumineuse exécutée par un segment donnée. Ces informations peuvent être extraites des colonnes MAX_SORT_SIZE et MAX_SORT_BLOCKS.

Exemple :

```
SQL> SELECT       max_sort_size, max_sort_blocks
  2  FROM   V$SORT_SEGMENT;
     MAX_SORT_S   MAX_SORT_B
     ----------   -----------
     25     442
```

3.2.2. Activité des segments temporaires

L'administrateur peut surveiller les utilisateurs qui se servent actuellement de segments temporaires en effectuant une jointure des vues V$SESSION et V$SORT_USAGE.
Voici une liste des colonnes de la vue V$SESSION et des données qu'elles contiennent :

- SADDR : Adresse de la session
- USERNAME : Nom de l'utilisateur qui effectue le tri
- SID : Identification de la session

Voici la liste partielle des colonnes de la vue V$SORT_USAGE et des données qu'elles contiennent :

- SESSION_ADDR : Adresse de la session
- TABLESPACE : type de tablespace dans lequel le segment temporaire est créé
- CONTENTS : type de contenu de la session
- EXTENTS : nombre d'extents utilisés par la session
- BLOCKS : nombre de blocks utilisés par la session
- USER : nom de l'utilisateur qui interroge cette vue (le nom de l'utilisateur qui effectue le tri s'obtient à partir de la vue V$SESSION)

Exemple :

```
SQL> SELECT s.username, u.contents, u.extents
  2  FROM   V$SESSION s, V$SORT_USAGE u
  3  WHERE  s.saddr = u.session_addr;
USERNAME    CONTENTS     EXTENTS
----------------------   -----------  ---------
SCOTT       TEMPORAY     4
```

→ Cette commande affiche les informations sur les tris actifs.

4. Gestion d'index

4.1. Types d'index

4.1.1. Classification des index

Un index est une arborescence permettant d'accéder directement à une ligne spécifique dans une table. Les index peuvent être classifiés en fonction de leur conception ou de leur implémentation physique.

Avec une classification logique, les index sont regroupés du point de vue de l'application. Les différents types d'index logiques sont les suivants :

- index à une colonne,
- index concaténé,
- index unique,
- index non unique.

Un index à une colonne comprend une seule colonne dans la clé d'index.
Par exemple, un index créé à partit de la colonne "empno" (numéro d'employé) de la table "emp" est un index à une colonne.

Un index concaténé, également appelé index composé, est créé à partit de plusieurs colonnes d'une table. Les colonnes d'in index concaténé ne doivent pas nécessairement être dans le même ordre que les colonnes de la table. En outre, ces colonnes ne sont pas forcément adjacentes.
Par exemple, un index créé à partir des colonnes "deptno" (numéro de département) et "job" (fonction) de la table "emp" est un index concaténé.
Un index concaténé peut contenir au maximum 32 colonnes. Toutefois, la taille cumulée de toutes les colonnes ne peut pas excéder un tiers de la taille du bloc de données.

Un index unique permet de s'assurer que deux lignes d'une table n'ont pas la même valeur dans la colonne qui définit l'index. Par conséquent, une clé d'index d'un index unique ne peut pointer que vers une seule ligne d'une table.

Dans un index non unique, une même clé peut avoir plusieurs lignes associées.

La classification physique des index dépend du format de stockage de ces index sur le disque. Les types d'index physique sont les suivants :

- index partitionné,
- index B-tree,
- index bitmap.

Un index partitionné permet de stocker les entrées d'index correspondant à un index dans plusieurs segments. Ce type d'index est créé pour des tables volumineuses car cela permet de diviser un index volumineux en plusieurs unités facilement gérables. Un index partitionné permet de diminuer a contention pour les recherches d'index. En effet, le partitionnement permet de répartir un index sur de nombreux tablespaces.
Les index partitionné sont souvent utilisés avec des tables partitionnées pour améliorer l'évolutivité et faciliter la gestion. Une partition d'index peut être créée pour chaque partition de table.

Un index B-tree est une arborescence ordonnée composée de nœuds d'index. Il contient une entrée par ligne.

Un index bitmap est un index physique contenant une entrée par groupe de lignes.

Le serveur Oracle gère tous les index lorsque des opérations DML sont exécutées sur une table. Les effets des différentes opérations DML sur les index B-tree sont les suivants :

Type d'opération DML	Effet sur les index B-tree
INSERT	Insertion de l'entrée d'index dans le bloc approprié
DELETE	Suppression logique de l'entrée d'index
UPDATE	Suppression logique de l'entrée d'index et insertion dans cet index

4.1.2. Index B-tree

Le processus d'E/S disque est le principal facteur déterminant la vitesse d'accès à une table. Pour réduire le temps d'E/S disque et améliorer les performances liées à l'accès aux tables, Oracle permet de créer des index de table. L'index B-tree est le type d'index le plus courant, et celui utilisé par défaut.
Un index B-tree stocke la valeur de clé et le ROWID de chaque ligne de la table. La structure d'un index B-tree comprend trois types de niveaux :

- la racine,
- les blocs de branche,
- les nœuds de feuille.

Au sommet d'un index B-tree se trouve la racine. Elle contient les entrées qui pointent vers le niveau suivant dans l'index.
Les blocs de branche constituent le niveau suivant d'un index B-tree. Ces blocs pointent vers les blocs du niveau suivant dans l'index.
Les nœuds de feuille constituent le niveau le plus bas d'un index B-tree. Ces nœuds contiennent les entrées d'index qui pointent vers les lignes d'une table. Les blocs feuille sont doublement liés pour que vous puissiez facilement parcourir l'index dans l'ordre croissant ou décroissant des valeurs de clé.

Une entrée d'index B-tree comporte les trois composants suivants :

- En-tête de l'entrée :
  Stocke le nombre de colonnes et les informations de verrouillage.
- Couples longueur de colonne / valeur de clé :
  Définit la taille d'une colonne dans une clé suivie de la valeur de la colonne.
- ROWID :
  Indique le ROWID de la ligne contenant les valeurs de clé.

Un index B-tree a plusieurs caractéristiques :

- Il contient une entrée pour chaque ligne de la table, mais il ne contient pas d'entrée pour une ligne affichant une valeur NULL dans toutes les colonnes clé d'index.
- Si plusieurs lignes contiennent la même valeur de clé, ces valeurs de clé sont répétées.
- Dans un index non partitionné, toutes les lignes appartiennent au même segment. Par conséquent, un ROWID réduit est utilisé pour pointer vers les lignes de la table.

4.1.3. Index à clé inversée

Lors de l'insertion d'enregistrements par ordre croissant de valeur de clé, par exemple par numéro d'employé généré par le système, des goulets d'étranglement d'E/S peuvent se produire dans l'index. En effet, toutes les mises à jour d'index ont lieu au même endroit dans l'arborescence de l'index. Pour éviter ces goulets d'étranglement, Oracle propose l'index à clé inversée.

Un index à clé inversée a plusieurs fonctions. Tout d'abord, il inverse les octets de chaque valeur de la colonne indexée, à l'exception du ROWID. Par exemple, si le numéro d'employé 7698 est inséré dans la table "emp", la valeur de clé 8967 ests stockée dans l'index.
Une autre fonction de l'index à clé inversée : il répartit les mises à jour d'index sur des valeurs de clé contiguës dans l'arborescence de l'index.

IL existe une restriction relative à l'utilisation d'un index à clé inversé. On ne peut pas effectuer un balayage de l'intervalle de valeurs indexées sur un index à clé inversée.

4.1.4. Index bitmap

Un index bitmap est organisé comme un index B-tree. Toutefois, le nœud de feuille stocke une bitmap pour chaque valeur de clé.
Chaque bit de la bitmap correspond à un ROWID possible. En outre, si un bit est défini, cela signifie que la ligne comportant le ROWID correspondant contient également la valeur de clé.

Les composants d'un index bitmap sont :

- l'en-tête de l'entrée :
  Nombre de colonnes et informations de verrouillage
- la valeur de clé :
  Couples longueur / valeur pour chaque colonne de clé
- le ROWID de début :
  ROWID correspondante au premier bit de la bitmap
- le ROWID de fin :
  ROWID correspondante au dernier bit de la bitmap
- le segment bitmap :
  Chaîne de bits définie quand la ligne correspondante contient la valeur de clé.

Si des modifications sont apportées à la colonne clé d'une table, les bitmaps doivent être modifiées. Pour cela, les segments bitmap appropriés sont verrouillés. Etant donné que les verrous sont placés sur l'intégralité du segment bitmap, une ligne couverte par le bitmap ne peut pas être mise à jour par d'autres transaction n'est pas terminée.

Un index bitmap peut être utilisé dans les situations suivantes :

- Lorsqu'une table contient des millions de lignes et que la colonne clé a une cardinalité faible.
- Lorsque les requêtes utilisent souvent une combinaison de la condition WHERE multiple avec l'opérateur OR.
- Lorsque les colonnes clé sont en lecture seule ou ne sont pas fréquemment mises à jour.

4.1.5. Index bitmap ou index B-tree

Oracle offre la possibilité d'utiliser un index B-tree ou un index bitmap. Pour déterminer l'index approprié en fonction des cas, il est indispensable de connaître les différences entre ces deux types d'index.

B-tree	Bitmap
Convient pour les colonnes à cardinalité élevée	Convient pour les colonnes à faible cardinalité
Les mises à jour sur les colonnes clé sont relativement légères.	Les mises à jour sur les colonnes clé sont très lourdes.
Ne convient pas aux requêtes qui utilisent le prédicat OR.	Convient aux requêtes qui utilisent le prédicat OR.
Utile dans un environnement OLTP.	Utile dans un environnement DSS.

4.2.Création d'index

4.2.1. Création d'un index B-tree

L'index B-tree est le type d'index le plus courant, et celui utilisé par défaut pour une table. Un index B-tree stocke des couples ROWID / valeur de clé. Ce type d'index peut être créé dans le compte de l'utilisateur auquel la table appartient ou dans un autre compte.
La syntaxe de la commande SQL permettant de créer un index B-tree est la suivante :

```
CREATE [UNIQUE] INDEX [schema.]index ON [schema.]table
(column [ASC|DESC][,column [ASC|DESC]]...)
[TABLESPACE tablespace_name]
[PCTFREE    integer]
[INITRANS   integer]
[MAXTRANS   interger]
[STORAGE    clause]
[LOGGING | NOLOGGING]
[NOSORT];
```

Exemple :

Créer un index B-tree normal EMP_IND1 appartenant au schéma SCOTT. Cet index doit être créé pour la table EMP dans le tablespace INDEX01 avec la valeur du paramètre PCTFREE à 25. Créer l'index à partir de la colonne EMPNO.

```
SQL> CREATE INDEX SCOTT.EMP_IND1 ON SCOTT.EMP(EMPNO)
  2  PCTFREE 25
  3  TABLESPACE INDEX01;
index created.
```

Pour créer un index de façon efficace, certaines règles doivent être prises en compte. Ces règles sont les suivantes :

- Trouver un compromis entre la requête et les besoins DML
- Placer l'index dans un tablespace distinct
- Utiliser des tailles d'extents uniformes
- Préférer l'option NOLOGGING pour les index volumineux
- Définir une valeur PCTFREE élevée si de nouvelles valeurs de clé doivent être ajoutées dans l'intervalle courant.

Un index peut être créé avec des propriétés semblables à celles d'un index existant avec l'outil Oracle Schema Manager.

4.2.2. Création d'un index à clé inversée

Pour éviter les goulets d'étranglement d'E/S, Oracle propose l'index à clé inversée. Un index à clé inversée inverse les octets de chaque colonne indexée.

```
CREATE [UNIQUE] INDEX [schema.]index ON [schema.]table
(column [ASC|DESC][,column [ASC|DESC]]...)
[TABLESPACE tablespace_name]
[PCTFREE    integer]
[INITRANS   integer]
[MAXTRANS   interger]
[STORAGE    clause]
[LOGGING | NOLOGGING]
REVERSE ;
```

Exemple :

Créer un index à clé inversée CUST_IND1 appartenant au schéma SCOTT et créé pour la table CUSTOMER. L'index est stocké dans le tablespace INDEX02 et est créé à partir de la colonne CUST_NO. La valeur du paramètre PCTFREE de l'index est de 30.

```
SQL> CREATE UNIQUE INDEX scott.cust_ind1
  2  ON scott.customer(cust_no) REVERSE
  3  PCTFREE 30
  4  TABLESPACE index02;
indexed created.
```

4.2.3. Création d'un index bitmap

Si les données impliquées ont une faible cardinalité, un index bitmap est plus approprié.

```
CREATE BITMAP INDEX [schema.]index ON [schema.]table
(column [ASC|DESC][,column [ASC|DESC]]...)
[TABLESPACE tablespace_name]
[PCTFREE    integer]
[INITRANS   integer]
[MAXTRANS   interger]
[STORAGE    clause]
[LOGGING | NOLOGGING]
[NOSORT];
```

Un index bitmap ne peut pas être unique.
Le paramètre d'initialisation CREATE_BITMAP_AREA_SIZE détermine la quantité d'espace utilisée pour stocker les segments bitmap dans la mémoire. La valeur par défaut de ce paramètre est 8 Mo. Une valeur supérieure à 8 Mo peut permettre d'accélérer la création de l'index.
Si la cardinalité est trop faible, l'administrateur peut affecter une valeur basse au paramètre CREATE_BITMAP_AREA_SIZE. Par exemple, si la cardinalité est seulement de deux, la valeur de ce paramètre peut être de l'ordre du kilo-octet plutôt que du mégaoctet. De manière générale, si la cardinalité est plus élevée, l'administrateur a besoin de davantage de mémoire pour obtenir des performances optimales.

Exemple :

Créer un index bitmap ORD_IND1 pour la table S_ORD sur la colonne ORDER_FILLED, stocké dans le tablespace INDEX01. La valeur du paramètre PCTFREE est de 25.

```
SQL> CREATE BITMAP INDEX SCOTT.ORD_IND1
  2  ON SCOTT.S_ORD (INV_NO)
  3  PCTFREE 25
  4  TABLESPACE INDEX01;
index created.
```

4.3.Gestion d'index

4.3.1. Modification des paramètres de stockage

Si un index vient à manquer d'espace de stockage, l'administrateur peut modifier les paramètres de stockage de cet index pour augmenter l'espace qui lui alloué.
Pour augmenter l'espace réservé à un index, l'administrateur peut augmenter le nombre maximal d'extents alloués à cet index.

Voici la syntaxe permettant de modifier les paramètres de stockage d'un index :

```
ALTER INDEX [schema.]index
[storage clause]
[INITRANS integer]
[MAXTRANS integer]
```

Exemple :

Pour augmenter l'espace disponible pour l'index EMP_IND1, il faut modifier ses paramètres de stockage en augmentant la valeur des paramètres MAXEXTENTS et MAXTRANS pour permettre à davantage d'extents et de transactions d'accéder à l'index simultanément.

```
SQL> ALTER INDEX SCOTT.EMP_IND1
  2  MAXTRANS 255
  3  STORAGE (MAXEXTENTS 200);
index altered.
```

4.3.2. Allocation d'espace dans l'index

Un administrateur peut être amené à ajouter des extents à un index en prévision d'une forte activité d'insertion sur une table. L'objet d'extents évite l'extension dynamique des index et la dégradation des performances qui en résulte.

La syntaxe permettant d'allouer de l'espace à un index est la suivante :

```
ALTER INDEX [schema.]index
ALLOCATE EXTENT ( [SIZE integer [ K | M ] ] [DATAFILE
'filename'] )
```

Exemple :

Ajouter des extents à l'index EMP_IND1 du schéma SCOTT en allouant de l'espace.

```
SQL> ALTER INDEX scott.emp_ind1
  2  ALLOCATE EXTENT (SIZE 200K);
index altered.
```

Cette commande affecte une taille de 200 K au nouvel extent de l'index EMP_IND1.

4.3.3. Libération dans l'index

De l'espace est libéré dans un index lorsque la table pour laquelle cet index a été créé est vidée. Pour limiter le gaspillage d'espace, il est possible de libérer manuellement l'espace inutilisé dans un index, au-dessus du high water mark.

La syntaxe permettant de libérer manuellement l'espace inutilisé dans un index au-dessus du high water mark est la suivante :

```
ALTER INDEX [schema.]index
[DATAFILE 'filename'] )
DEALLOCATE UNUSED [KEEP integer [ K | M ] ]
```

Exemple :

Libérer l'espace inutilisé au-dessus du high water mark dans l'index EMP_IND1 du schéma SCOTT.

```
SQL> ALTER INDEX scott.emp_ind1
  2  DEALLOCATE UNUSED;
index altered.
```

4.3.4. Situations de reconstruction d'un index

Une gestion efficace des index permet d'extraire plus rapidement les données souhaitées à partir d'un objet. La reconstruction des index est nécessaire à une gestion efficace des index.
Les index sont reconstruits dans différentes situations :

- Pour déplacer un index vers un autre tablespace
  Cette opération peut être nécessaire si l'index se trouve dans le même tablespace que la table ou si des objets doivent être répartis sur plusieurs disques.
- Pour optimiser l'utilisation de l'espace en enlevant les entrées supprimées.
  Ce problème se pose généralement pour les index en biais.
  Exemple : un index a été créé à partir de la colonne ORDER_NO de la table ORDERS, dans laquelle les commandes exécutées sont supprimées et les nouvelles commandes sont ajoutées. Si quelques commandes anciennes sont encore en cours, la table peut contenir plusieurs blocs de feuille d'index comportant des entrées supprimées. La reconstruction de l'index permet alors d'améliorer l'utilisation de l'espace.
- Pour convertir in index à clé inversée existant en index B-tree normal ou inversement.

La commande permettant de reconstruire un index est la suivante :

```
ALTER INDEX [schema.]index REBUILD
[TABLESPACE tablespace]
[PCTFREE integer]
[INITRANS integer]
[MAXTRANS integer]
[storage clause]
[LOGGING | NOLOGGING]
[REVERSE | NOREVERSE];
```

Cette commande ne peut pas être utilisée pour convertir un index bitmap en index B-tree, ou vice-versa.

Exemple :

Reconstruire l'index EMP_IND1 appartenant au schéma SCOTT dans le tablespace INDEX01.

```
SQL> ALTER INDEX scott.emp_ind1 REBUILD
  2  TABLESPACE index01;
index altered.
```

4.3.5. Caractéristiques de reconstruction d'un index

Lorsque plusieurs modifications sont apportées aux données d'un index existant, cet index peut être soit supprimé puis recréé, soit reconstruit. La reconstruction d'un index prend moins de temps puisqu'une partie des données existantes est déjà triée. Le processus permettant de reconstruire un index est appelé reconstruction d'index.

Une reconstruction d'index permet de créer un nouvel index en utilisant un index existant comme source de données.
Une reconstruction d'index a certaines caractéristiques.

- Elle évite les opérations de tri car l'index est construit à partir d'un index existant. Cela permet d'obtenir de meilleures performances.

- L'ancien index est supprimé une fois la construction du nouvel index terminée. Lors d'une reconstruction d'index, il faut disposer de suffisamment d'espace pour stocker l'ancien et le nouvel index dans leurs tablespaces respectifs.
- Un index reconstruit ne contient pas d'éléments supprimés. Cela permet donc une utilisation plus efficace de l'espace.
- Lors d'une reconstruction d'index, les requêtes peuvent continuer à utiliser l'index existant pendant la construction du nouvel index.

4.3.6. Contrôle de la validité d'un index

Pour vérifier si un index est valide, tous ses blocs peuvent être analysés afin de s'assurer qu'ils ne sont pas corrompus. Cette opération met à jour la vue INDEX_STATS pour prendre en compte les informations relatives à l'index.
Ainsi, l'administrateur pourra réorganiser un index si il constate qu'il comporte beaucoup de lignes supprimées. Par exemple, un index peut être reconstruit si la proportion de colonnes DEL_LF_ROWS par rapport aux colonnes LF_ROWS excède 30 pour cent.

Avant d'interroger la vue IDEX_STATS, il faut la remplir.
Voici la commande SQL utilisée pour remplir cette vue :

```
ANALYSE INDEX [schéma.]index VALIDATE STRUCTURE;
```

Exemple :

Extraire les informations des colonnes NAME, BLOCKS, LF_ROWS et DEL_LF_ROWS concernant les index

```
SQL> SELECT name, blocks,lf_rows,del_lf_rows
  2  FROM INDEX_STATS;
NAME        BLOCKS LF_ROWS      DEL_LF_ROWS
------------------      ---------    ------------ ------------
EMP_IND1         5     14     0
```

4.3.7. Suppression d'un index

Les index qui ne sont utilisés que ponctuellement ne nécessitent pas d'être conservés. C'est le cas notamment d'un index créé pour un système OLTP dans lequel des requêtes ad hoc sont générées toues les ans ou tous les trimestres pour recueillir des informations en vue d'une réunion.

Lors de l'échec d'une instance Oracle, des index peuvent être marqués INVALID pour certains types d'opérations comme les chargements. Ces index peuvent être supprimées et recréés.

Un index peut être supprimé avant le chargement d'une grande quantité de données, puis recrée. Cela permet d'améliorer les performances du chargement et d'utiliser l'espace plus efficacement dans l'index.

La syntaxe de la commande SQL permettant de supprimer un index est la suivante :

```
DROP INDEX [schéma.]index;
```

4.3.8. Informations sur les index

Les index de la base de données peuvent être surveillé en extrayant des informations générales sur ces index ou des informations relatives aux colonnes indexées à partir des vues DBA_INDEXES et DBA_IND_COLUMNS.

Un liste partielle des colonnes contenues dans la vue de dictionnaire de données DBA_INDEXES :
- OWNER

- INDEX_NAME
- INDEX_TYPE
- TABLE_OWNER
- TABLE_NAME
- UNIQUENESS
- TABLESPACE_NAME
- LOGGIG
- STATUS

Exemple :

Extraire des informations générales sur les index : interroger la vue DBA_INDEXES pour extraire les informations des colonnes INDEX_NAME, TABLESPACE_NAME et INDEX_TYPE pour tous les index appartenant au schéma SCOTT.

```
SQL> SELECT       index_name, tablespace_name, index_type
  2  FROM   dba_indexes
  3  WHERE  owner = 'SCOTT';
INDEX_NAME                     TABLESPACE_NAME           INDEX_TYPE
------------------------------ ------------------------- -----------
PK_DEPT                        USER_DATA                 NORMAL
PK_EMP                         USER_DATA                 NORMAL
S_CUSTOMER_ID_PK               USER_DATA                 NORMAL
S_DEPT_ID_PK                          USER_DATA
     NORMAL
S_DEPT_NAMPE_REGION_ID_UK      USER_DATA                 NORMAL
```

Une liste partielle des colonnes contenues dans la vue de dictionnaire de données DBA_IND_COLUMNS :

- INDEX_OWNER
- INDEX_NAME
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- COLUMN_POSITION
- COLUMN_LENGTH

Exemple :

Extraire des informations sur les colonnes indexées : interroger la vue DBA_IND_COLUMNS pour extraire les colonnes INDEX_NAME, TABLE_OWNER et TABLE_NAME. Rechercher les informations sur les index appartenant à SCOTT et les trier en fonction de la colonne INDEX_NAME.

```
SQL> SELECT       index_name, table_owner, table_name
  2  FROM   dba_ind_columns
  3  WHERE  index_owner = 'SCOTT'
  4  ORDER BY     index_name;
INDEX_NAME                     TABLE_OWNER               INDEX_TYPE
------------------------------ ------------------------- -----------
PK_DEPT                        SCOTT                     DEPT
PK_EMP                         SCOTT                     EMP
S_CUSTOMER_ID_PK               SCOTT                     S_CUSTOMER
S_DEPT_ID_PK                          SCOTT
     S_DEPT
S_DEPT_NAMPE_REGION_ID_UK      SCOTT                     S_DEPT
```