Gérer les objets de schéma



ORACLE

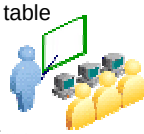
Riadh ZAAFRANI

2ème années SI - 2ème années SI

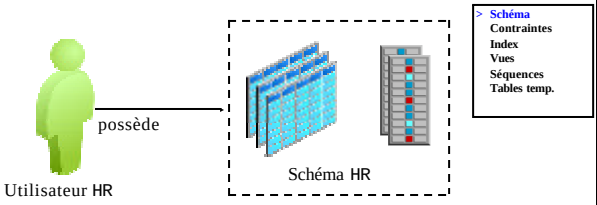
Avril 2010

Objectifs du cours

- A la fin de ce cours, vous pourrez :
  - définir des objets de schéma et des types de données
  - créer et modifier des tables
  - définir des contraintes
  - afficher les colonnes et le contenu d'une table
  - créer des index
  - créer des vues
  - créer des séquences
  - expliquer l'utilisation des tables temporaires
  - utiliser le dictionnaire de données



Qu'est-ce qu'un schéma ?



- Un **schéma** est un ensemble d'objets de base de données appartenant à un **utilisateur particulier**.
- Généralement, dans le cadre d'une base de données de production, le terme "utilisateur" ne désigne pas une personne mais une application. Un schéma porte le même nom que l'utilisateur auquel il appartient.

Qu'est-ce qu'un schéma ?

- Les objets de schéma sont les **structures logiques** qui font directement référence aux données de la base. Ils incluent des structures telles que les tables, les vues et les index.
- **Remarque :** Il n'est pas nécessaire qu'un schéma soit directement lié à un tablespace unique. Vous pouvez définir des configurations dans lesquelles les objets d'un même schéma se trouvent dans des tablespaces différents, ou dans lesquelles un tablespace contient des objets issus de différents schémas.

Qu'est-ce qu'un schéma ?

- Lorsque vous créez la base de données, plusieurs schémas sont créés automatiquement. Deux d'entre eux sont particulièrement importants :
  - **Schéma SYS :** Il contient le dictionnaire de données.
  - **Schéma SYSTEM :** Il contient des tables et des vues supplémentaires dans lesquelles sont stockées des informations d'administration.
- Lors de l'installation complète d'une base de données Oracle, différents exemples de schémas liés entre eux sont aussi installés automatiquement. Ces exemples peuvent avoir différents niveaux de complexité.

Qu'est-ce qu'un schéma ?

- Les schémas fournis sont les suivants :
- **HR :** Le schéma Human Resources est un schéma simple utilisé pour présenter les notions de base de ce cours.
- **OE :** Le schéma Order Entry permet de traiter de sujets présentant une complexité moyenne. Une multitude de types de données y sont disponibles. Le sous-schéma **OC** (Online Catalog) est un ensemble d'objets de base de données relationnelle objet créé dans le schéma **OE**.
- **PM :** Le schéma Product Media est dédié aux types de données multimédias.
- **QS :** Le schéma Queued Shipping contient un ensemble de schémas utilisés pour illustrer les fonctionnalités d'Oracle Advanced Queuing.
- **SH :** Le schéma Sales History permet les démonstrations impliquant des quantités importantes de données. Une extension de ce schéma fournit la prise en charge du traitement analytique avancé.

Accéder aux objets de schéma

- Vous pouvez accéder rapidement à de nombreux types d'objet de schéma à partir de la région **Schema** de la page Database **Administration**. Lorsque vous cliquez sur l'un des liens, la page **Résultas** apparaît.



Accéder aux objets de schéma

- Dans la région **Rechercher** de la page, vous pouvez entrer un nom de schéma et un nom d'objet afin de rechercher un objet spécifique. En outre, vous pouvez rechercher d'autres types d'objets via la région **Rechercher**, en effectuant des sélections dans la liste déroulante. Cette liste inclut des types d'objet supplémentaires qui ne sont pas présentés sous forme de liens dans la page Database Administration.



Nommer les objets de base de données

- La longueur des noms doit être comprise entre 1 et 30 octets, à deux exceptions près :
  - Les noms de base de données sont limités à 8 octets.
  - Les noms de lien de base de données peuvent comporter jusqu'à 128 octets.
- Les noms qui ne sont pas inclus entre guillemets ne peuvent pas être des mots réservés Oracle.
- Les noms qui ne sont pas inclus entre guillemets doivent commencer par un caractère alphabétique du jeu de caractères de la base de données.



Nommer les objets de base de données

- Lorsque vous indiquez le nom d'un objet de base de données, vous pouvez l'inclure entre des guillemets (""). En procédant ainsi, vous pouvez contourner plusieurs des règles d'appellation indiquées dans la diapositive précédente. Toutefois, cette pratique est déconseillée car si vous nommez un objet de cette manière, vous devez toujours y faire référence en faisant figurer le nom entre guillemets. Par exemple, si vous nommez une table "Local Temp", vous devez procéder de la façon suivante :

```
SQL> select * from "Local Temp";
TEMP_DATE    LO_TEMP    HI_TEMP
---------- ---------- ----------
01-DEC-03         30         41
```

Nommer les objets de base de données

- Si vous entrez le nom en ne respectant pas les majuscules et les minuscules, vous obtenez une erreur :

```
SQL> select * from "local temp";
select * from "local temp"
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

- Les noms qui ne sont pas inclus entre guillemets sont stockés en majuscules et ne sont pas sensibles à la casse. Lorsqu'une instruction SQL est traitée, les noms qui ne sont pas inclus entre guillemets sont convertis en majuscules.

Nommer les objets de base de données

- Les **identificateurs** qui ne sont pas inclus entre guillemets peuvent uniquement contenir des caractères alphanumériques issus du jeu de caractères de la base de données, ainsi que le caractère de soulignement **(_)**, le symbole dollar **($)** et le symbole dièse **(#)**. Les liens de base de données peuvent en outre contenir des points **(.)** et le symbole "arobas" **(@)**. Il est vivement déconseillé d'utiliser les symboles **$** et **#** dans les identificateurs qui ne sont pas inclus entre guillemets.
- Les identificateurs inclus entre guillemets peuvent contenir n'importe quel caractère ou signe de ponctuation, ainsi que des espaces. Toutefois, ni les identificateurs qui sont inclus entre guillemets, ni ceux qui ne le sont pas ne peuvent contenir de guillemets.

Définir des types de données pour les tables

- Types de données courants :
  - **CHAR(*size* [BYTE|CHAR]) :** chaîne de caractères de **longueur fixe**, dont la taille est de *size* octets ou caractères. La taille maximale est de 2 000 octets ou caractères, tandis que la taille minimale est de 1 octet (valeur par défaut).
  - **VARCHAR2(*size*) :** chaîne de caractères de **longueur variable**, dont la taille maximale est de *size* octets ou caractères. La taille maximale est de 4 000 octets. Vous devez impérativement indiquer la taille du type de données VARCHAR2.
  - **DATE :** date valide comprise entre le 1er janvier 4712 avant JC et le 31 décembre 9999 après JC. Ce type de données contient également l'heure (en heures, minutes et secondes).
  - **NUMBER :** nombre dont la précision est *p* et l'échelle *s*. La précision peut être comprise entre 1 et 38, et l'échelle entre -84 et 127.



Définir des types de données pour les tables

- **BINARY_FLOAT :** nombre en virgule flottante sur 32 bits. Ce type de données requiert 5 octets, en comptant l'octet de longueur.
- **BINARY_DOUBLE :** nombre en virgule flottante sur 64 bits. Ce type de données requiert 9 octets.
- **FLOAT(*p*) :** type de données ANSI (American National Standards Institute). Le type de données FLOAT est un nombre en virgule flottante dont la précision binaire est *p*. La précision par défaut de ce type de données est de 126 en binaire, soit 38 en décimal.
- **INTEGER :** type de données équivalent au type **NUMBER(p,0)**.
- **NCHAR(*length*) :** type de données Unicode exclusivement. Lorsque vous créez une table comportant une colonne **NCHAR**, vous définissez la longueur de la colonne en nombre de caractères. Vous définissez le jeu de caractères nationaux lors de la création de la base de données. La taille maximale autorisée pour une colonne est de 2000 octets. Si vous insérez une valeur dont la longueur est inférieure à celle de la colonne, la base de données Oracle lui ajoute des espaces pour qu'elle atteigne la longueur totale de la colonne. Vous ne pouvez pas insérer une valeur **CHAR** dans une colonne **NCHAR**, ni insérer une valeur **NCHAR** dans une colonne **CHAR**.

Définir des types de données pour les tables

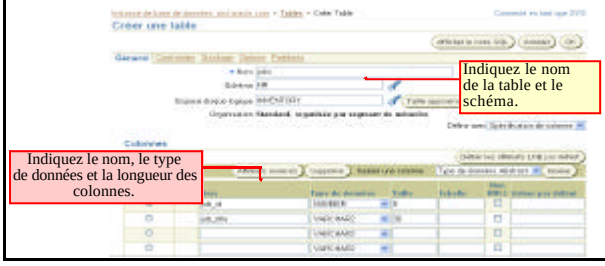
- **NVARCHAR2(size [BYTE|CHAR]) :** type de données Unicode exclusivement. Il est semblable au type de données **NCHAR**, à ceci près que sa longueur maximale est de 4 000 octets et que le serveur Oracle n'ajoute pas d'espaces pour compléter la longueur.
- **RAW(*size*) :** données binaires raw, dont la longueur est de ***size*** octets. La taille maximale est de 2 000 octets. Vous devez obligatoirement définir la taille d'une valeur RAW.
- **ROWID :** chaîne en base 64, représentant l'adresse unique d'une ligne dans la table. Ce type de données est destiné essentiellement aux valeurs renvoyées par la pseudo-colonne **ROWID**.
- **UROWID :** chaîne en base 64, représentant l'adresse logique d'une ligne d'une table organisée en index. La taille (facultative) est celle d'une colonne de type UROWID. La taille maximale (valeur par défaut) est de 4000 octets.

Définir des types de données pour les tables

- **BLOB :** objet binaire de grande taille.
- **CLOB :** objet de type caractère de grande taille, contenant des caractères mono-octets ou multi-octets. Les jeux de caractères de longueur fixe et de longueur variable sont pris en charge. Ils utilisent tous le jeu de caractères CHAR de la base de données.
- **NCLOB :** objet de type caractère de grande taille, contenant des caractères Unicode.
- **BFILE :** type de données contenant un pointeur vers un fichier binaire de grande taille stocké en-dehors de la base. Il permet un accès d'E/S de type flux d'octets aux objets LOB externes résidant sur le serveur de base de données. La taille maximale est de 4 gigaoctets.
- **TIMESTAMP(*fractional_seconds_precision*) :** type de données permettant d'indiquer les valeurs année/mois/jour de la date, ainsi que les valeurs heure/minute/seconde de l'heure, où ***fractional_seconds_precision*** correspond au nombre de chiffres avec lesquels sont exprimées les fractions de seconde. Les valeurs acceptées sont comprises entre 0 et 9 (la valeur par défaut étant 6).

Créer et modifier des tables

- Pour créer une table à l'aide d'Enterprise Manager, effectuez les opérations suivantes :
- 1. Cliquez sur **Tables** dans la région **Schéma** de la page **Administration**. La page Tables apparaît.



Créer et modifier des tables

- 2. Si vous connaissez le nom du schéma, entrez tout ou partie de ce nom dans le champ Schéma de la région **Rechercher**. Si vous ne connaissez pas le nom du schéma, cliquez sur l'icône représentant une torche, en regard du champ Schema. La fenêtre **Rechercher et sélectionner : Schéma** apparaît. Vous pouvez parcourir les noms des schémas et sélectionner celui que vous recherchez.
- 3. Cliquez sur **Créer**. La page **Créer une table : Organisation de la table** apparaît.
- 4. Acceptez la valeur par défaut **(Standard, organisée par segment de mémoire)** et cliquez sur **Continuer**. La page **Créer une table** apparaît.
- 5. Entrez le nom de la table dans le champ **Nom**.

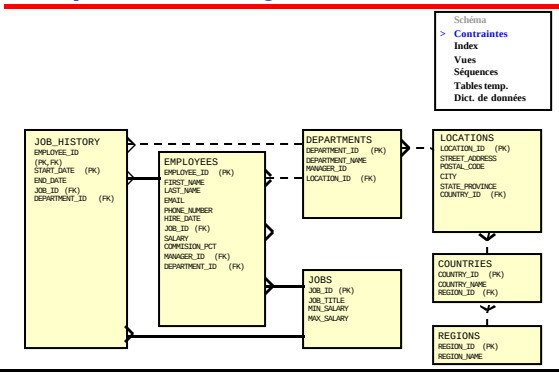
Créer et modifier des tables

- 6. Entrez le nom du schéma dans le champ **Schéma**, ou cliquez sur l'icône représentant une torche afin d'appeler la fonction de recherche.
- 7. Entrez le nom du tablespace dans le champ **Espace disque logique**, ou cliquez sur l'icône représentant une torche afin d'appeler la fonction de recherche.
- 8. Dans la région **Colonnes**, entrez le nom et le type de données des colonnes.
- 9. Cliquez sur **OK**. Un message de mise à jour apparaît, indiquant que la table a été créée avec succès.

Créer et modifier des tables

- Vous pouvez modifier une table avec Enterprise Manager, en suivant la procédure ci-dessous. Dans l'exemple considéré, une colonne est ajoutée à une table.
- 1. Dans la page **Tables**, sélectionnez la table dans la liste des résultats, puis cliquez sur **Modifier**.
- 2. Dans la page **Modifier une table**, cliquez sur le bouton **Ajouter 5 colonnes à la table**. Une liste modifiable de colonnes apparaît.
- 3. Entrez le nom, le type de données et la taille de la nouvelle colonne.
- 4. Cliquez sur **Appliquer**. Un message de mise à jour apparaît, indiquant que la table a été modifiée avec succès.

Comprendre l'intégrité des données



Comprendre l'intégrité des données

- Vous pouvez utiliser les contraintes d'intégrité suivantes pour imposer des restrictions concernant la saisie des valeurs dans les colonnes :
- **`NOT NULL` :** Par défaut, toutes les colonnes d'une table autorisent les valeurs **`NULL`**. La valeur `NULL` signifie l'absence de valeur. Une contrainte `NOT NULL` impose qu'une colonne d'une table ne contienne aucune valeur `NULL`. Par exemple, vous pouvez définir une contrainte `NOT NULL` afin d'imposer la saisie d'une valeur dans la colonne `LAST_NAME` pour chaque ligne de la table `EMPLOYEES`.
- **Clé `UNIQUE` :** Une contrainte d'intégrité de clé **`UNIQUE`** impose que chaque valeur d'une colonne ou d'un ensemble de colonnes (clé) soit unique (deux lignes d'une table ne doivent pas comporter de valeurs en double). Par exemple, une contrainte `UNIQUE` est définie sur la colonne `DEPARTMENT_NAME` de la table `DEPARTMENTS` afin d'interdire que plusieurs lignes comportent le même nom de département. Sauf cas particulier, cette contrainte est appliquée par le biais d'un index unique.

Comprendre l'intégrité des données

- **`PRIMARY KEY` :** Une table de la base de données ne peut pas comporter plus d'une contrainte **`PRIMARY KEY`**. Les valeurs du groupe de colonnes soumis à cette contrainte constituent l'identificateur unique de la ligne. En effet, chaque ligne est nommée par ses valeurs de clé primaire.
- L'implémentation de la contrainte d'intégrité `PRIMARY KEY` par le serveur Oracle garantit que les deux conditions suivantes sont vraies :
  - Deux lignes d'une table ne comportent pas de valeurs en double dans la colonne ou l'ensemble de colonnes désigné.
  - Les colonnes de clé primaire n'autorisent pas les valeurs `NULL`. Autrement dit, une valeur doit être présente dans les colonnes de clé primaire de chaque ligne.
- En règle générale, la base de données applique les contraintes PRIMARY KEY par le biais d'index. La contrainte de clé primaire créée pour la colonne DEPARTMENT_ID de la table DEPARTMENTS est mise en oeuvre par la création implicite des éléments suivants :
  - un index unique sur cette colonne,
  - une contrainte `NOT NULL` pour cette colonne.

Comprendre l'intégrité des données

- **Contraintes d'intégrité référentielle :** Différentes tables d'une base de données relationnelle peuvent être liées par l'intermédiaire de colonnes communes. Les règles qui régissent les relations entre les colonnes doivent être respectées. Les règles d'intégrité référentielle garantissent que ces relations sont préservées.
- Une contrainte d'intégrité référentielle impose que pour chaque ligne d'une table, la valeur de la clé étrangère corresponde à une valeur d'une clé parent.
- Par exemple, une clé étrangère est définie sur la colonne `DEPARTMENT_ID` de la table `EMPLOYEES`. Elle permet de s'assurer que chaque valeur de cette colonne correspond à une valeur de la clé primaire de la table `DEPARTMENTS`. Par conséquent, il ne peut exister aucun numéro de département erroné dans la colonne `DEPARTMENT_ID` de la table `DEPARTMENTS`.
- Il existe un autre type de contrainte d'intégrité référentielle, appelé contrainte d'intégrité auto-référentielle. Ce type de clé étrangère fait référence à une clé parent qui se trouve dans la même table.

Comprendre l'intégrité des données

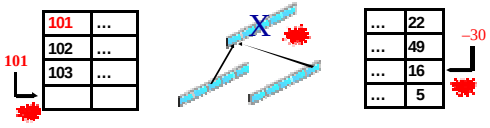
- **Contraintes CHECK :** Une contrainte d'intégrité `CHECK` appliquée sur une colonne ou un ensemble de colonnes impose qu'une condition particulière soit vraie ou inconnue pour chaque ligne de la table.
- Si une instruction LMD (Langage de manipulation de données) entraîne une valeur `false` pour l'évaluation de la contrainte `CHECK`, l'instruction est annulée (rollback).

Définir des contraintes

- Pour ajouter une contrainte à une table à l'aide d'Enterprise Manager, effectuez les opérations suivantes :
- 1. Sélectionnez la table dans la page **Tables**, puis cliquez sur **Modifier**.
- 2. Cliquez sur **Contraintes**. Vous accédez à la page **Contraintes**, qui indique toutes les contraintes définies sur la table.
- 3. Dans la liste déroulante, sélectionnez le type de contrainte à ajouter, puis cliquez sur **Ajouter**.
- 4. Entrez les informations appropriées concernant le type de contrainte que vous définissez. Cliquez sur **OK**.

Violations de contrainte

- Voici quelques exemples de violations de contrainte :
  - Insertion d'une valeur de clé primaire en double
  - Suppression du parent d'une ligne enfant dans une contrainte d'intégrité référentielle
  - Mise à jour d'une colonne en lui affectant une valeur qui se trouve en-dehors des limites d'une contrainte CHECK

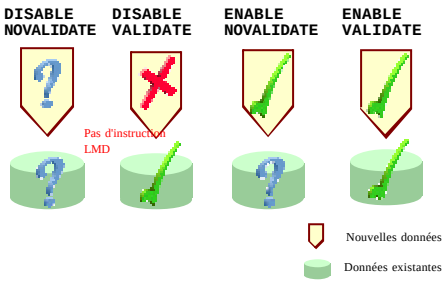


Etats possibles d'une contrainte

- Pour pouvoir gérer les situations dans lesquelles des données doivent enfreindre temporairement une contrainte, vous pouvez appliquer différents états à une contrainte.
- Une contrainte d'intégrité peut être activée (**ENABLE**) ou désactivée (**DISABLE**). Lorsqu'elle est activée, les données sont vérifiées au fur et à mesure de leur saisie ou de leur mise à jour dans la base. Les données qui ne respectent pas la règle de la contrainte ne sont pas acceptées.
- Lorsqu'une contrainte est désactivée, il est possible d'entrer dans la base des données non conformes à cette contrainte. Vous pouvez appliquer les états suivants à une contrainte d'intégrité :

```
DISABLE NOVALIDATE
DISABLE VALIDATE
ENABLE NOVALIDATE
ENABLE VALIDATE
```

Etats possibles d'une contrainte



Etats possibles d'une contrainte

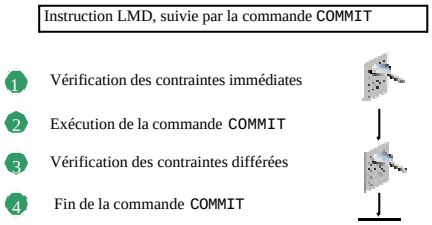
- **DISABLE NOVALIDATE :** Les données ne sont pas vérifiées, qu'elles soient nouvelles ou existantes. Elles peuvent ne pas être conformes à la contrainte. Cet état est souvent utilisé lorsque les données proviennent d'une source déjà validée et que la table est en lecture seule (aucune nouvelle donnée n'est entrée dans la table).
- **DISABLE VALIDATE :** Lorsqu'une contrainte se trouve dans cet état, il n'est pas possible de modifier les colonnes auxquelles elle s'applique. En effet, il serait illogique d'avoir validé les données existantes, puis d'autoriser la saisie de données non vérifiées dans la table. Cet état est souvent utilisé lorsque les données existantes doivent faire l'objet d'une validation mais ne seront pas modifiées ultérieurement, et que le recours à l'index n'est pas nécessaire pour assurer de bonnes performances.

Etats possibles d'une contrainte

- **ENABLE NOVALIDATE :** Les nouvelles données doivent être conformes à la contrainte mais les données existantes se trouvent dans un état inconnu. Cet état est fréquemment utilisé pour permettre la correction des violations de contrainte existantes tout en empêchant de nouvelles violations de se produire dans le système.
- **ENABLE VALIDATE :** Toutes les données doivent être conformes à la contrainte, qu'elles soient nouvelles ou existantes. Il s'agit de l'état standard d'une contrainte (état par défaut).

Vérification des contraintes

- La vérification des contraintes s'effectue :
  - lors de l'exécution de l'instruction pour les contraintes ***immédiates***
  - lors de l'exécution de la commande **COMMIT** pour les contraintes ***différées***



Vérification des contraintes

- Les **contraintes immédiates**, ou contraintes non différées, sont appliquées à la fin de chaque instruction LMD (Langage de manipulation de données). Une violation de contrainte entraîne l'annulation (rollback) de l'instruction. Si une contrainte produit une action telle que **delete cascade**, l'action est considérée comme faisant partie de l'instruction qui l'a provoquée. Vous ne pouvez pas transformer une contrainte immédiate en contrainte pouvant être différée.

Vérification des contraintes

- Les **contraintes différées** ne sont vérifiées que lorsqu'une transaction est validée (commit). Si une violation de contrainte est détectée lors de la phase de validation, l'intégralité de la transaction est annulée (rollback). Ce type de contrainte s'avère particulièrement utile lors de la saisie simultanée de lignes parent et de lignes enfant liées par une relation de clé étrangère. C'est le cas, par exemple, d'un système de gestion des commandes client, dans lequel la commande et les articles concernés sont entrés en même temps.
- Vous pouvez appliquer l'un des paramètres suivants à une contrainte pouvant être différée :
  - **Initially immediate** indique que, par défaut, la contrainte doit être traitée comme une contrainte immédiate, sauf indication contraire explicite.
  - **Initially deferred** stipule que, par défaut, la contrainte ne doit être appliquée qu'à la fin de la transaction.

Créer des contraintes via des instructions SQL : Exemples

a

```
ALTER TABLE countries
ADD (UNIQUE(country_name) ENABLE NOVALIDATE);
```

- Après l'exécution de cette instruction, toute insertion ou mise à jour effectuée dans la table `COUNTRIES` doit comporter une valeur `COUNTRY_NAME` unique.
- Toutefois, il se peut que la table contienne déjà des valeurs `COUNTRY_NAME` non uniques.
- Le mot-clé `NOVALIDATE` ordonne d'ignorer ces valeurs en double. La contrainte ne s'applique qu'aux nouvelles lignes.

Créer des contraintes via des instructions SQL : Exemples

b

```
ALTER TABLE employees
ADD CONSTRAINT pk PRIMARY KEY (employee_id)
```

- Cette instruction ajoute une clé primaire à la table `EMPLOYEE`. Le nom de la contrainte est **PK** et la clé primaire correspond à la colonne `EMPLOYEE_ID`.

Créer des contraintes via des instructions SQL : Exemples

```
CREATE TABLE t1 (pk NUMBER PRIMARY KEY, fk NUMBER, c1
NUMBER, c 2 NUMBER,
CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t 1,
CONSTRAINT ck1 CHECK (pk > 0 and c 1 > 0));
```

- Cette instruction définit les contraintes au moment de la création de la table, plutôt que d'utiliser ultérieurement une instruction ALTER TABLE. La contrainte RI impose aux valeurs de la colonne FK d'être présentes dans la colonne de clé primaire de la table T1. La contrainte CK1 stipule que les colonnes PK et C1 doivent contenir une valeur supérieure à zéro.
- **Remarque :** Chaque contrainte est dotée d'un nom. Si vous ne l'indiquez pas dans l'instruction LDD (Langage de définition de données), un nom fourni pas le système et commençant par SYS_ est affecté.

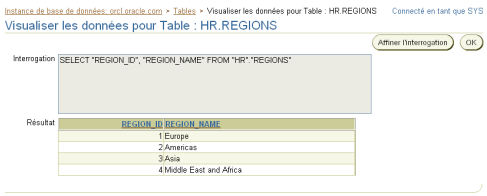
Afficher les colonnes d'une table

- Pour afficher les attributs d'une table à l'aide d'Enterprise Manager, effectuez les opérations suivantes, Sélectionnez une table et cliquez sur le bouton **Visualiser** afin d'afficher les attributs de cette table.



Afficher le contenu d'une table

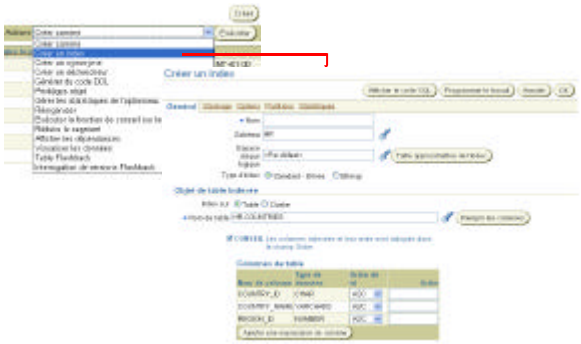
- Pour afficher les lignes d'une table à l'aide d'Enterprise Manager, Sélectionnez la table dans la page Tables puis Sélectionnez **Visualiser les données** dans le menu Actions, puis cliquez sur **Exécuter**.



Afficher le contenu d'une table

- La page **Visualiser les données pour Table** apparaît.
- Les données des lignes de la table s'affichent dans la région **Résultat**.
- La zone **Interrogation(Query)** affiche l'interrogation SQL qui est exécutée pour générer les résultats.
- Dans cette page, vous pouvez cliquer sur le nom de n'importe quelle colonne et trier les données de cette colonne par ordre croissant ou décroissant.
- Si vous souhaitez modifier l'interrogation, cliquez sur le bouton **Affiner l'interrogation(Refine Query)**.
- Dans la page **Affiner l'interrogation pour Table**, vous pouvez sélectionner les colonnes que vous souhaitez afficher et indiquer une clause WHERE dans l'instruction SQL afin de limiter les résultats.

Actions sur les tables



Actions sur les tables

- Vous pouvez sélectionner une table, puis effectuer des actions sur cette table. Vous pouvez notamment effectuer les actions suivantes :
- **Créer comme :** Cette action permet de créer une table présentant la même structure que la table sélectionnée. Vous devez modifier le nom des contraintes. Vous pouvez ajouter ou supprimer des colonnes, et apporter d'autres modifications à la structure de la table avant de créer cette dernière.
- **Créer un index :** Utilisez cette option pour créer des index sur une table.
- **Générer du code DDL :** Cette action génère l'instruction LDD (Langage de définition de données) qui représente la table telle qu'elle existe. Cette instruction peut ensuite être copiée dans un fichier texte afin d'être utilisée en tant que script ou à des fins de documentation.

Actions sur les tables

- **Privilèges objet :** Par défaut, lorsqu'une table est créée, seul son propriétaire peut l'utiliser. Le propriétaire doit octroyer des privilèges aux autres utilisateurs pour qu'ils puissent exécuter des instructions LMD ou LDD sur la table.
- **Afficher les dépendances :** Cette opération affiche les objets dont dépend la table, ou les objets qui dépendent de la table.
- **Visualiser les données :** Cette action permet de sélectionner et d'afficher les données de la table en mode lecture seule.

Supprimer une table

- **Syntaxe : DROP TABLE [schema.] table [CASCADE CONSTRAINTS] [PURGE]**

```
DROP TABLE hr.employees PURGE;
```

- La commande **DROP TABLE** table entraîne la suppression des éléments suivants :
  - Données
  - Structure de la table
  - Déclencheurs de base de données
  - Index correspondants
  - Privilèges objet associés
- En l'absence de la clause **PURGE**, la définition de la table, les index associés et les déclencheurs (triggers) sont placés dans la **corbeille**. Les données de la table sont conservées, mais elles sont inaccessibles sans la définition de la table. Lorsque vous supprimez une table via **Enterprise Manager**, la clause PURGE n'est pas utilisée.

Supprimer une table

- La commande **FLASHBACK TABLE** permet de récupérer les objets de schéma dans la corbeille.
- La commande **PURGE RECYCLEBIN** vide la corbeille.
- L'option **CASCADE CONSTRAINTS** permet de supprimer toutes les contraintes d'intégrité référentielle dépendantes.
- **Remarque :** Si vous n'utilisez pas l'option **PURGE**, l'espace occupé par la table et ses index entre toujours en compte dans le calcul du quota autorisé pour l'utilisateur dans les tablespaces concernés. En d'autres termes, l'espace reste considéré comme étant utilisé.

Vider une table

- **Syntaxe : TRUNCATE TABLE [schema.] table [{DROP| REUSE} STORAGE]**

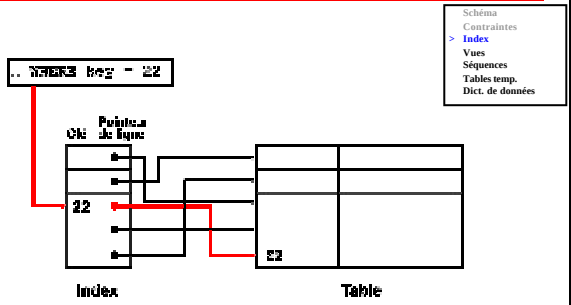
```
TRUNCATE TABLE hr.employees;
```

- **TRUNCATE TABLE** produit les effets suivants :
- La table est marquée comme étant vide. En effet, le repère **high-water mark (HWM)** est placé au début de la table, de sorte que les lignes de celle-ci ne sont plus disponibles.
- Aucune donnée d'annulation (undo) n'est générée et la commande procède implicitement à une validation (commit) car **TRUNCATE TABLE** est une commande LDD (Langage de définition de données).
- Les index correspondants sont également vidés.
- Une table à laquelle une clé étrangère fait actuellement référence ne peut pas être vidée.

Vider une table

- Les déclencheurs (triggers) de suppression ne sont pas exécutés lors de l'utilisation de cette commande.
- La commande **TRUNCATE** s'avère généralement beaucoup plus rapide qu'une instruction **DELETE** pour supprimer toutes les lignes de la table, pour les raisons suivantes :
  - La base de données Oracle réinitialise le repère high-water mark de la table au lieu d'appliquer l'opération DELETE à chaque ligne.
  - Aucune donnée d'annulation n'est générée.

Index



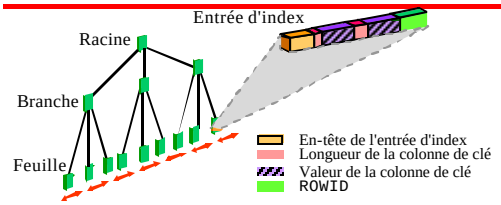
Index

- Les **index** sont des structures facultatives associées aux tables. Ils peuvent être créés afin d'améliorer les performances de mise à jour et d'extraction des données. Un index Oracle fournit un chemin d'accès direct vers une ligne de données.
- Les **index** peuvent être créés sur une ou plusieurs colonnes d'une table. Dès lors qu'un index créé, il est automatiquement tenu à jour et utilisé par le serveur Oracle. Les mises à jour des données d'une table, telles que l'ajout, la mise à jour ou la suppression de lignes, sont automatiquement propagées vers tous les index concernés, de manière totalement transparente pour l'utilisateur.

Types d'index

- Les types d'index les plus courants sont les suivants :
  - **B-tree**
  - **Bitmap**
- Les valeurs de clé d'un **index B-tree** sont stockées dans une arborescence équilibrée **(Balanced tree - B-tree)**, ce qui permet d'effectuer des recherches binaires rapides.
- Un **index bitmap** inclut un bitmap pour chaque valeur de clé indexée. Chaque bit du bitmap correspond à une ligne de la table indexée. Cela permet d'effectuer des recherches rapides lorsqu'il existe un nombre restreint de valeurs distinctes **(on dit que la colonne indexée présente une faible cardinalité)**. Par exemple l'indicateur du sexe d'une personne ne peut comporter que les valeurs "M" et "F". La recherche ne porte donc que sur deux bitmaps. En revanche, pour une colonne `phone_number`, il faudrait gérer et explorer un si grand nombre de bitmaps que l'opération deviendrait totalement inefficace.

Structure d'un index B-tree



- Au sommet de l'index se trouve la **racine**, qui contient les entrées pointant vers le niveau suivant de l'index. Le niveau suivant comprend les blocs **branche**, qui pointent vers les blocs du niveau suivant de l'index. Au plus bas niveau se trouvent les noeuds **feuille**, qui contiennent les entrées d'index pointant vers les lignes de la table. Les blocs feuille font l'objet d'une liaison double, afin de faciliter le balayage de l'index dans l'ordre croissant ou décroissant des valeurs de clé.

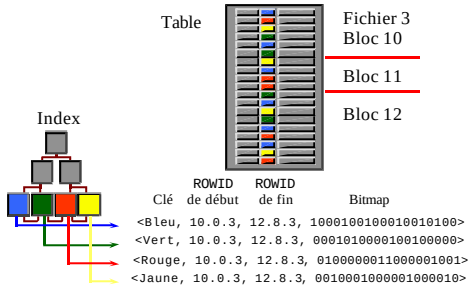
Format des entrées d'index B-Tree

- Une entrée d'index est composée des éléments suivants :
  - Un **en-tête**, dans lequel sont stockés le nombre de colonnes et les informations de verrouillage.
  - Des paires **longueur-valeur** qui définissent la taille d'une colonne de clé suivie de la valeur de la colonne (le nombre de paires de ce type ne doit pas excéder le nombre de colonnes présentes dans l'index).
  - Le **`ROWID`** d'une ligne qui contient les valeurs de clé.

index bitmap

- Il est plus judicieux d'utiliser des **index bitmap** que des index B-tree dans les cas suivants :
  - Lorsqu'une table contient des millions de lignes et que les colonnes de clé présentent une faible cardinalité (c'est-à-dire que chaque colonne de clé contient un nombre très restreint de valeurs distinctes). Par exemple, les index bitmap peuvent être préférables aux index B-tree pour les colonnes concernant le sexe des personnes et leur situation familiale dans une table qui contient des enregistrements de passeports.
  - Lorsque les interrogations utilisent souvent une combinaison de plusieurs conditions `WHERE` incluant l'opérateur `OR`.
  - Lorsque les colonnes de clé ne sont utilisées qu'en lecture ou font rarement l'objet de mises à jour.

Structure d'un index bitmap



Structure d'un index bitmap

- Un **index bitmap** est organisé comme un index B-tree, à ceci près que les noeuds feuille stockent un **bitmap** pour chaque valeur de clé et non une liste de **ROWID**. Chaque bit du bitmap correspond à un ROWID possible. Le bit a la valeur 1 lorsque la ligne correspondante contient la valeur de clé.
- Un noeud feuille d'un index bitmap contient les éléments suivants :
  - Un **en-tête**, qui contient le nombre de colonnes et les informations de verrouillage.
  - Des valeurs de clé consistant en des paires **longueur-valeur** pour chaque colonne de clé. Dans l'exemple de la diapositive, la clé est une colonne unique, et la première entrée comporte la valeur de clé Bleu.

Structure d'un index bitmap

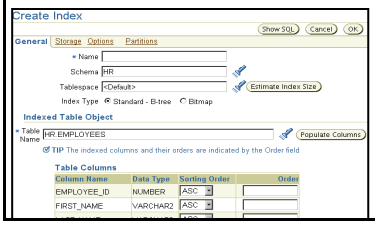
- Un **ROWID de début**. Dans l'exemple considéré, il correspond au numéro de bloc dix, au numéro de ligne zéro et au numéro de fichier trois.
- Un **ROWID de fin**. Dans l'exemple, il correspond au numéro de bloc douze, au numéro de ligne huit et au numéro de fichier trois.
- Un **segment bitmap** composé d'une chaîne de bits. (Un bit est égal à 1 lorsque la ligne correspondante contient la valeur de clé. Sinon, il est égal à 0. Le serveur Oracle utilise une technique de compression brevetée pour le stockage des segments bitmap.)
- Le ROWID de début est le ROWID de la première ligne vers laquelle pointe le segment bitmap. Autrement dit, le premier bit du bitmap correspond à ce ROWID, le deuxième bit correspond à la ligne suivante du bloc et le ROWID de fin pointe vers la dernière ligne de la table incluse dans le segment bitmap. Les index bitmap utilisent des ROWID restreints.

Options relatives aux index

- Un **index unique** permet de garantir que chaque valeur indexée est unique.
- Les valeurs de clé d'un index peuvent être stockées dans l'ordre croissant ou décroissant.
- Un **index composé** est basé sur plusieurs colonnes :
  CREATE INDEX name_ix ON employees (last_name, first_name);
- Un **index basé sur une fonction** repose sur la valeur renvoyée par une fonction :
  CREATE INDEX tree_vol_ix ON
  TREES(volume(species,height,circumference));
- Un **index compressé** supprime les valeurs de clé en double. Il est utilisé pour réduire la consommation d'espace disque lors de la phase d'exécution.

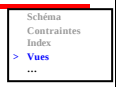
Créer des index

- Vous pouvez cliquer sur le lien **Index** sous l'en-tête **Schéma** de la page **Administration** afin d'afficher la page Index. Vous pouvez afficher les attributs des index ou utiliser le menu Actions pour visualiser les dépendances d'un index.
- Les index peuvent être créés explicitement, ou implicitement par l'intermédiaire de contraintes placées sur une table. Un index créé implicitement peut résulter, par exemple, de la définition d'une clé primaire. Un index unique est alors créé automatiquement pour la mise en oeuvre de l'unicité des valeurs de la colonne.




```
CREATE INDEX my_index
ON
employees(last_name,
first_name);
```

Qu'est-ce qu'une vue ?



- Les **vues** sont des représentations personnalisées de données provenant d'une ou de plusieurs tables, ou d'autres vues.
- Elles peuvent être considérées comme des interrogations stockées car elles peuvent masquer des conditions et des jointures très complexes, ainsi que d'autres expressions et structures SQL complexes.
- Une vue ne contient pas de données à proprement parler. Ses données sont issues des tables sur lesquelles elle est basée. Ces tables sont appelées tables de base de la vue.

Qu'est-ce qu'une vue ?

Table LOCATION

LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROVINCE	CO
2200	12-98 Victoria Street	2901	Sydney	New South Wales	AU
2800	Rua Frei Caneca 1360	01307-002	Sao Paulo	Sao Paulo	BR
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT

Table COUNTRY

CO	COUNTRY_NAME	REGION_ID
AR	Argentina	2
AU	Australia	3
BE	Belgium	1
BR	Brazil	2

Vue

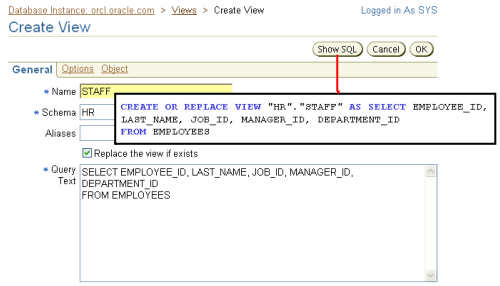
LOCATION_ID	COUNTRY_NAME
2200	Australia
2800	Brazil

```
CREATE VIEW v AS SELECT location_id, country_name
FROM locations l, countries c
WHERE l.country_id = c.country_id
AND c.country_id in ('AU','BR');
```

Créer des vues

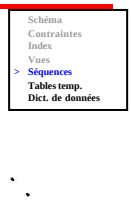
- A l'instar des tables, les vues peuvent faire l'objet d'interrogations, de mises à jour, d'insertions et de suppressions, avec cependant quelques restrictions. Toutes les opérations effectuées sur une vue affectent les tables de base de cette vue. Les vues fournissent un niveau de sécurité supplémentaire en limitant l'accès à un ensemble prédéterminé de lignes et de colonnes d'une table. Elles masquent également la complexité des données et permettent le stockage d'interrogations complexes.
- Pour afficher les vues définies dans la base de données, cliquez sur le lien **Vues** sous l'en-tête **Schéma** de la page **Administration**.

Créer des vues



Séquences

- Une séquence est un mécanisme de génération automatique de nombres entiers selon un modèle spécifique.
  - Une séquence comporte un nom, qui est utilisé pour la demande de la valeur suivante.
  - Une séquence n'est associée à aucune table ou colonne particulière.
  - La progression peut s'effectuer de manière croissante ou décroissante.
  - L'intervalle entre les numéros peut être de n'importe quelle taille.
  - Une séquence peut effectuer une boucle lorsqu'une limite est atteinte.



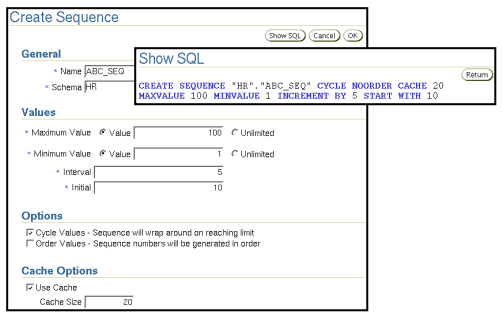
Créer une séquence

- Vous pouvez afficher et créer des séquences à l'aide d'Enterprise Manager, en cliquant sur le lien **Séquences** sous l'en-tête **Schéma** de la page **Administration**. Voici un récapitulatif des options de création de séquence disponibles :
- **Nom :** Nom de la séquence, utilisé pour faire référence à la séquence.
- **Schéma :** Propriétaire de la séquence.
- **Valeur maximale :** Indiquez la valeur maximale que la séquence peut générer. L'option Unlimited indique une valeur maximale de 10^{27} pour une séquence croissante et de –1 pour une séquence décroissante. La valeur par défaut est Unlimited.
- **Valeur minimale :** Indiquez la valeur minimale de la séquence. Elle doit être inférieure ou égale à Initial, et strictement inférieure à valeur maximale.

Créer une séquence

- **Intervalle :** Indique l'intervalle entre les numéros de séquence.
- **Initial :** Indique le premier numéro de séquence à générer.
- **Valeurs de cycle :** Lorsqu'une séquence croissante atteint sa valeur maximale, elle poursuit en générant à nouveau sa valeur minimale. Lorsqu'une séquence décroissante atteint sa valeur minimale, elle poursuit en générant à nouveau sa valeur maximale.
- **Valeurs de tri :** Cette option garantit que les numéros de séquence sont générés dans l'ordre dans lequel ils sont demandés.
- **Options de cache :** Indique le nombre de numéros de séquence que la base de données Oracle préalloue et conserve en mémoire pour un accès plus rapide.

Créer une séquence

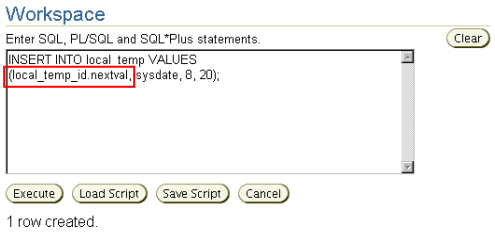


Utiliser une séquence

- Pour faire référence à des valeurs de séquence dans des instructions SQL, utilisez les pseudo-colonnes suivantes :
- **CURRVAL** : renvoie la valeur actuelle d'une séquence.
- **NEXTVAL** : incrémente la séquence et renvoie la valeur suivante.
- Vous devez qualifier CURRVAL et NEXTVAL avec le nom de la séquence :

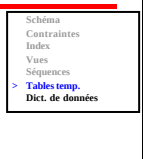
  ***sequence*.CURRVAL**
  ***sequence*.NEXTVAL**
- La première référence à NEXTVAL renvoie la valeur initiale de la séquence. Les références suivantes à NEXTVAL incrémentent la valeur de la séquence du nombre indiqué et renvoient la nouvelle valeur. Une référence à CURRVAL renvoie toujours la valeur actuelle de la séquence, qui est la valeur renvoyée par la dernière référence à NEXTVAL.

Utiliser une séquence

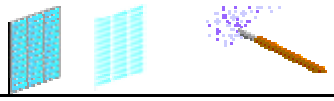


Tables temporaires

- Vous pouvez utiliser des tables temporaires lorsque vous avez besoin de stocker des données de manière privée pour la réalisation d'une tâche, et que vous souhaitez supprimer ces données une fois la tâche terminée, à la fin d'une transaction ou d'une session.



- Les tables temporaires vous évitent d'avoir à masquer vos données pour les autres sessions et à supprimer les données générées lorsque vous avez terminé. Les seules données d'une table temporaire visibles par une session sont celles que la session a inséré.

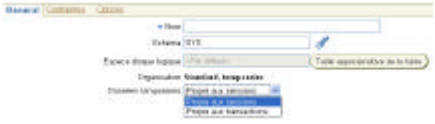


Tables temporaires

- Une table temporaire peut être propre à une transaction ou à une session. Dans le cas des tables temporaires propres à une transaction, les données sont présentes pendant toute la durée de la transaction. Pour les tables temporaires propres à une session, elles sont présentes pendant toute la durée de la session. Les clauses suivantes contrôlent la durée de vie des lignes :
  - **ON COMMIT DELETE ROWS :** Indique que la durée de vie des lignes insérées correspond à la durée de la transaction uniquement.
  - **ON COMMIT PRESERVE ROWS :** Indique que la durée de vie des lignes insérées correspond à la durée de la session

Tables temporaires

- Pour créer des tables temporaires à l'aide d'Enterprise Manager, cliquez sur l'option **Temporaire**, dans la page **Créer une table : Organisation de la table**. Cliquez sur **Continuer**. La page qui apparaît vous permet de préciser si la table temporaire est propre à la session ou à une transaction.



- Le champ Tablespace est désactivé car une table temporaire est toujours créée dans le tablespace temporaire de l'utilisateur. Aucun autre tablespace ne peut être indiqué.

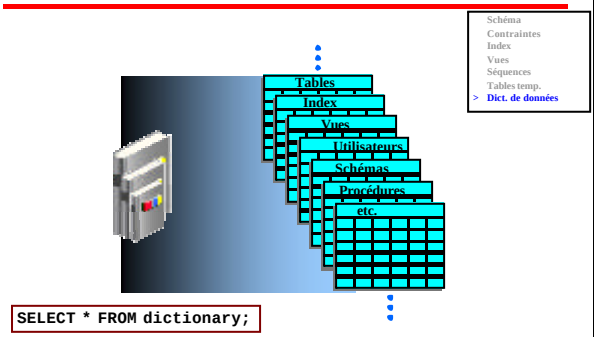
Tables temporaires : Utilisation

- Utilisez la clause **GLOBAL TEMPORARY** pour créer une table temporaire :

```
CREATE GLOBAL TEMPORARY TABLE employees_temp
ON COMMIT PRESERVE ROWS
AS SELECT * FROM employees;
```

- Utilisez la la commande **TRUNCATE TABLE** pour supprimer le contenu de la table. Cette opération supprime toutes les données que vous avez insérées. Elle s'avère plus efficace que la commande **DELETE**.
- Vous pouvez créer les éléments suivants sur une table temporaire : Index, Vues et Déclencheurs.

Dictionnaire de données : Présentation




```
SELECT * FROM dictionary;
```

Dictionnaire de données : Présentation

- Le **dictionnaire de données** d'Oracle constitue la description d'une base de données. Il contient le nom et les attributs de tous les objets de la base. Lors de la création ou de la modification d'un objet, il est mis à jour afin de refléter les changements opérés.
- Les informations du dictionnaire sont stockées dans les **tables** de base gérées par la base de données Oracle. Vous ne lisez pas directement ces tables mais vous y accédez par le biais de **vues prédéfinies**.

Dictionnaire de données : Présentation

- Le **dictionnaire de données** :
  - est utilisé par le serveur de base de données Oracle pour la recherche d'informations sur les utilisateurs, les objets, les contraintes et le stockage,
  - est mis à jour par le serveur de base de données Oracle lorsque des définitions ou des structures d'objet sont modifiées,
  - peut être utilisé par n'importe quel utilisateur pour la recherche d'informations sur la base de données,
  - appartient à l'utilisateur **SYS**,
  - ne doit jamais être modifié directement à l'aide d'une instruction SQL.

Dictionnaire de données : Présentation

- La vue **DICTIONARY** du dictionnaire de données, ou le synonyme **DICT** correspondant, contient le nom et la description de tous les éléments inclus dans le dictionnaire de données.
- Utilisez la vue **DICT_COLUMNS** pour afficher les colonnes des vues et leur définition.p

Vues du dictionnaire de données

- Chaque nom de vue a un **préfixe** qui indique le type et la quantité de données qu'un utilisateur est autorisé à voir.
- La vue globale, identifiée par le préfixe **DBA_**, n'est accessible qu'aux utilisateurs disposant de privilèges DBA.
- Le niveau suivant de privilège est déterminé par le préfixe **ALL_**, qui représente tous les objets que l'utilisateur exécutant l'interrogation est autorisé à voir, qu'il en soit le propriétaire ou non. Par exemple, si l'utilisateur **USER_A** est autorisé à accéder à une table dont **USER_B** est le propriétaire, **USER_A** voit cette table dans toutes les vues **ALL_** concernant les noms de table.
- Le préfixe **USER_** est associé aux vues les plus restreintes. Ces vues affichent seulement les objets dont l'utilisateur exécutant l'interrogation est le propriétaire, c'est-à-dire les objets présents dans le schéma de l'utilisateur.

Vues du dictionnaire de données

	Personnes autorisées à interroger	Contenu	Sous-ensemble de	Remarques
DBA_	DBA	Tout	N/A	Peut comporter des colonnes supplémentaires destinées au DBA uniquement
ALL_	Tous	Tous les éléments que l'utilisateur est autorisé à voir	Vues DBA_	Inclut les objets propres à l'utilisateur
USER_	Tous	Tous les éléments dont l'utilisateur est le propriétaire	Vues ALL_	Généralement identique à ALL_, hormis la colonne OWNER manquante. Certaines vues sont dotées de noms abrégés comme synonymes PUBLIC.

Vues du dictionnaire de données

- D'une manière générale, chaque ensemble de vues est un sous-ensemble de l'ensemble de vues nécessitant des privilèges supérieurs, au niveau des lignes et des colonnes.
- Toutefois, les vues d'un ensemble donné n'ont pas toutes leur pendant dans les autres ensembles de vues. Par exemple, il existe une vue **DBA_LOCK** mais pas de vue **ALL_LOCK**. En effet, seul un administrateur de base de données (DBA) est intéressé par les données relatives aux verrous externes (locks).
- Les vues **DBA_** peuvent être interrogées par les utilisateurs disposant du privilège **SYSDBA** ou **SELECT ANY DICTIONARY**.

Dictionnaire de données : Exemples d'utilisation

- a - Quel est le nom des tables qui ont été créées dans votre schéma (et quel est le nom du tablespace dans lequel elles résident) ?

a)
```
SELECT table_name, tablespace_name FROM
user_tables;
```

- b - Quelles informations importantes relatives aux séquences de la base de données avez-vous accès ?

b)
```
SELECT sequence_name, min_value, max_value,
increment_by FROM all_sequences WHERE
sequence_owner IN ('MDSYS','XDB');
```

Dictionnaire de données : Exemples d'utilisation

- c - Quels utilisateurs de la base de données sont autorisés actuellement à se connecter ?

c)
```
SELECT USERNAME, ACCOUNT_STATUS FROM
dba_users WHERE ACCOUNT_STATUS = 'OPEN';
```

- Quelles sont les colonnes de la vue DBA_INDEXES ?

d)
```
DESCRIBE dba_indexes;
```

- Cette interrogation permet de connaître les informations que vous pouvez consulter au sujet de tous les index de la base de données. Voici un exemple de sortie partielle de l'exécution de cette commande :

OWNER	NOT NULL	VARCHAR2(30)
INDEX_NAME	NOT NULL	VARCHAR2(30)
INDEX_TYPE		VARCHAR2(27)
TABLE_OWNER	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)

Synthèse



- Ce cours vous a permis d'apprendre à :
  - définir des objets de schéma et des types de données
  - créer et modifier des tables
  - définir des contraintes
  - afficher les colonnes et le contenu d'une table
  - créer des index
  - créer des vues
  - créer des séquences
  - expliquer l'utilisation des tables temporaires
  - utiliser le dictionnaire de données