Bases de données

Jean-Yves Antoine

VALORIA - Université François Rabelais

Jean-Yves.Antoine@univ-tours.fr

L3 S&T mention Informatique

Bases de données

SGBD ORACLE

Organisation du Dictionnaire Oracle

L3 S&T mention Informatique

Dictionnaire Oracle

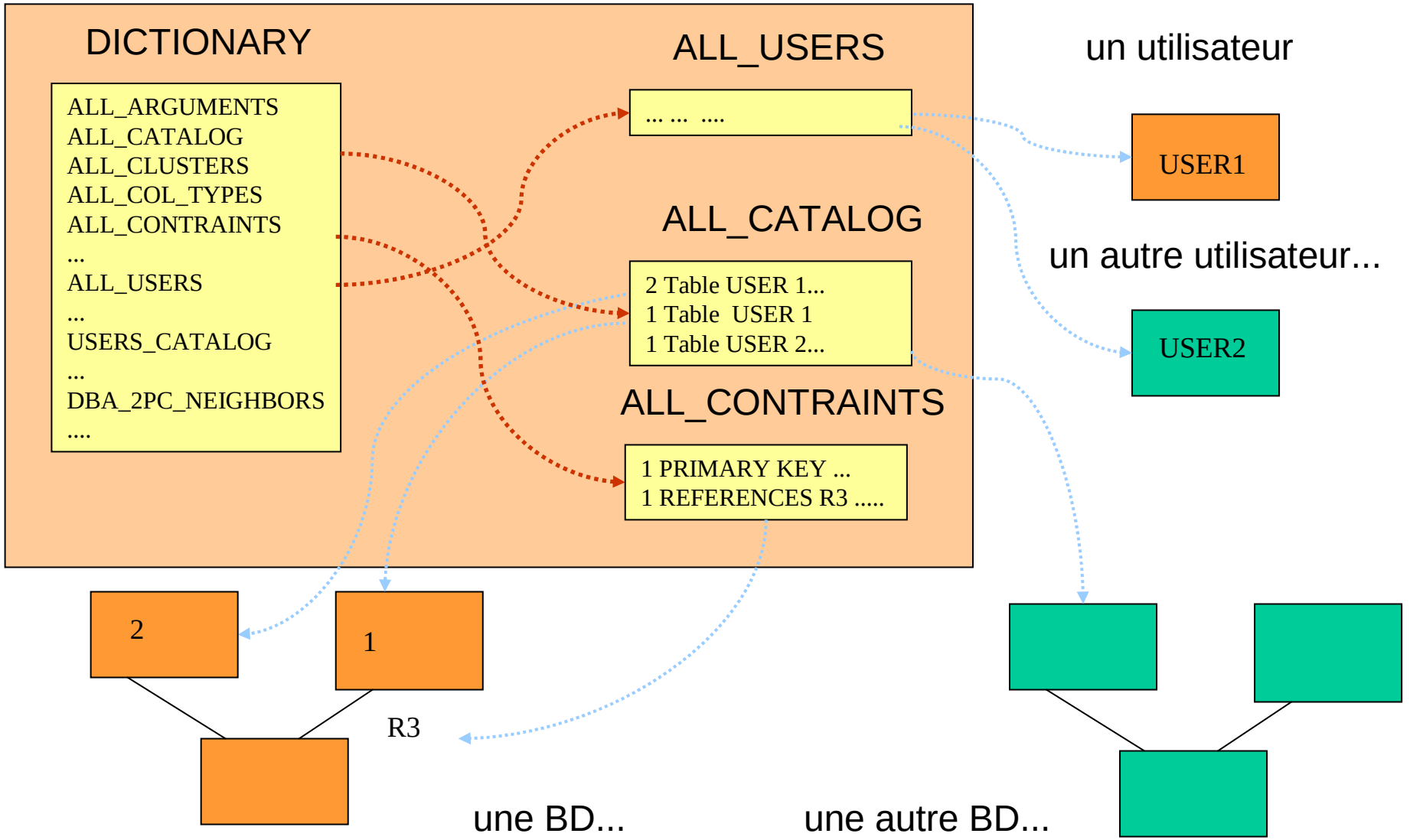
- **Base de données interne gérant tous les objets** (tables, vues, utilisateurs, index etc...) **connus du SGBD**
- **Information cryptée accessible uniquement via des vues**

  - **USER_*** vues relatives aux objets appartement à l'utilisateur
  - **ALL_*** vues relatives aux objets accessibles par l'utilisateur (ceux dont il est propriétaire ainsi que ceux pour lesquels il a un droit d'accès)
  - **DBA_*** vues relatives à l'administration (droits utilisateurs, *rollback segments*, etc...) de la BD. Accessibles uniquement par l'utilisateur SYSTEM.
  - **V$*** vues dynamiques relatives aux suivi des performances

- **Vue DICTIONARY**

  Point d'entrée sur le dictionnaire : vue décrivant tous les objets du dictionnaire

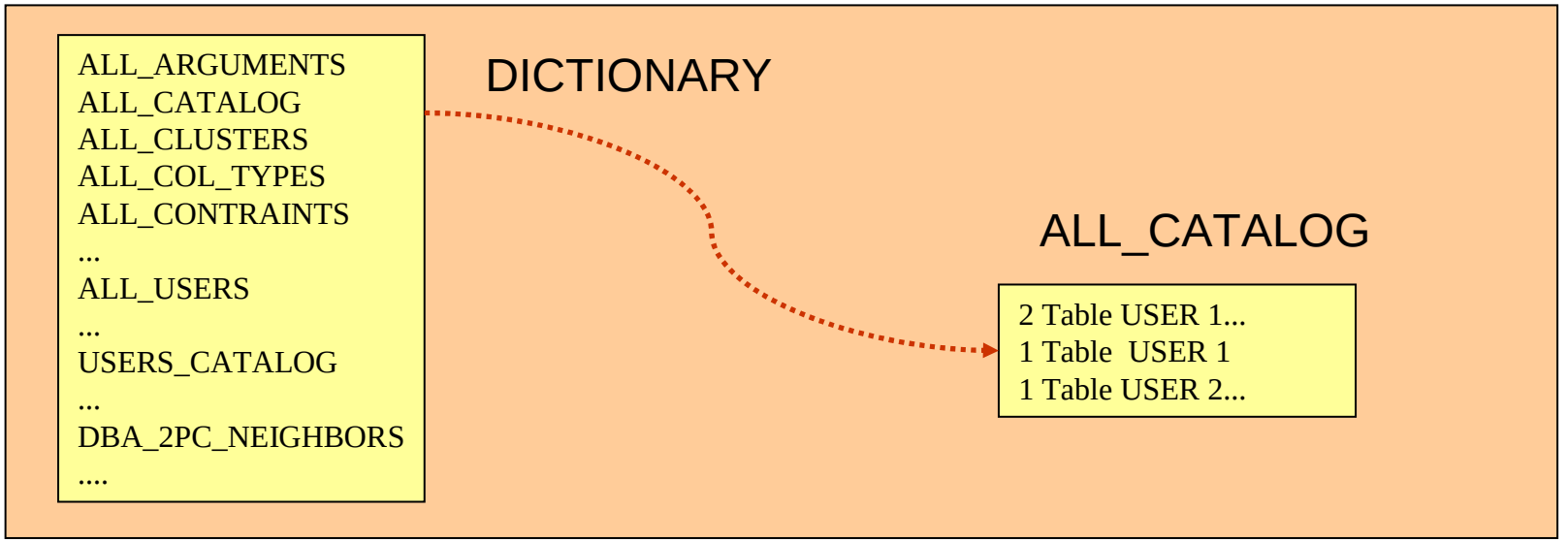
Dictionnaire Oracle



Dictionnaire Oracle

Vues principales (seules étudiées en L3)

- **DICTIONARY** Table de référence de toutes les vues du dictionnaire
- **ALL_CATALOG** Tous les objets (tables, vues,...) accessibles à l'utilisateur
- **USER_CATALOG** Tous les objets qui sont propriétés de l'utilisateur, avec leur type



Objets Oracle (seuls rencontrés en L3)

- **Tables** - **Vues**
- **Synonyme** Alias sur un objet. Objet permanent sauvegardé dans le dictionnaire Oracle

Bases de données

SQL et LDD

Langage de Définition de Données

L3 S&T Mention Informatique

Création de relation : CREATE TABLE

- **Création de relation** : définition du schéma de la relation
  - commande CREATE TABLE
  - définition des attributs de la relation
  - définition des contraintes associées aux attributs ou à la relation
- **Contraintes de relation**
  - **recommandé** : contraintes d'intégrité référentielle et de clé primaire
  - **obligatoire** : contrainte sur plusieurs attributs (exemple : unicité sur un couple d'attributs)
- **Contraintes d'attributs**
  - **obligatoire** : domaine de l'attribut (contrainte implicite dans le CREATE TABLE)
  - **recommandé** : autres cas non cités précédemment

Syntaxe CREATE TABLE

- **Syntaxe simplifiée**

  CREATE TABLE <nom table>
  (<ATTRIBUTS> [, <CONTRAINTES TABLE >]) ;

- **Définition attribut**

  <ATTRIBUTS> ::= <DEF ATTR> [, <ATTRIBUTS>]

  <DEF ATTR> ::= <nom attribut> <DATATYPE> [<CONTRAINTES ATTR>]

- **DATATYPE : principaux types de données**

CHAR(n)	Chaîne de caractères de longueur fixe égale à n
VARCHAR(n)	Chaîne de caractères de longueur maximale égale à n
NUMBER(n,p)	Nombre réel de précision n et d ’échelle p
NUMBER	Entier (Réel à 0 décimales)
DATE	Date jj/mm/aaaa
LONG	Données de type caractère allant jusqu’à 2 Go.

CREATE TABLE : contraintes

- **Type de contraintes**

Domaine	contrainte implicite : DATATYPE
Unicité	clause SQL : UNIQUE
Renseignement	clause SQL : NOT NULL
Arithmétique et Logique	clause SQL : CHECK
Clé primaire	clause SQL : PRIMARY KEY
Référentielle	clause SQL : FOREIGN KEY ... REFERENCES

- **Clause CONSTRAINT**
  - Une commande par type (`table_constraint, column_constraint`)
  - S'il n'est pas obligatoire de définir un nom à chaque contrainte, cela est fortement conseillé pour une observation facile dans le dictionnaire (ALL_CONSTRAINTS).

CREATE TABLE : contraintes

- **Contraintes d'attribut** (conseillées)

  <CONTRAINTES_ATTR> ::= <CONTR_ATTR> [, < CONTR_ATTR > ...]

  < CONTR_ATTR > ::= CONSTRAINT <nom_contrainte> NOT NULL

  < CONTR_ATTR > ::= CONSTRAINT <nom_contrainte> CHECK (<condition>)

  < CONTR_ATTR > ::= CONSTRAINT <nom_contrainte> UNIQUE

- **Contraintes de relation** (conseillées)

  <CONTRAINTES_TABLE> ::= <CONTR> [, < CONTR > ...] }

  < CONTR> ::= CONSTRAINT <nom_contrainte> PRIMARY KEY (<liste_attr>)

  < CONTR> ::= CONSTRAINT <nom_contrainte> FOREIGN KEY (<liste_attr>)

  REFERENCES <table_source>((<liste_attr>)

  [ON DELETE CASCADE]

  < CONTR> ::= CONSTRAINT <nom_contrainte> UNIQUE (<liste_attr>)

  < CONTR> ::= CONSTRAINT <nom_contrainte> CHECK (<condition sur plusieurs attributs>)

CREATE TABLE : exemple

```
CREATE TABLE personne
(
  num_h       NUMBER
              CONSTRAINT pos_num_h CHECK (num_h > 0),
  nom         VARCHAR(30)
              CONSTRAINT nn_personne_nom NOT NULL,
  prenom      VARCHAR(20)
              CONSTRAINT nn_personne_prenom NOT NULL,
  parti       NUMBER,
  CONSTRAINT pk_personne_numh PRIMARY KEY(num_h),
  CONSTRAINT fk_personne_parti FOREIGN KEY(parti)
              REFERENCES parti(num_p)
);
```

- **Syntaxe détaillée** : consulter la documentation

ALTER TABLE

- **Utilisation**

  Modification de la structure d'une relation (ajout d'attribut, de contrainte) si l 'utilisateur dispose de tels droits

ADD	ajout d'une contrainte ou d'un attribut
MODIFY	modification d'une contrainte ou d'un attribut
DROP	suppression d'une contrainte

- **Syntaxe** ALTER TABLE <nom_table>

  [ADD | MODIFY | DROP] (<clause de définition ou suppression>) ;

```
ALTER TABLE personne
ADD( CONSTRAINT pers_nom_pre_uni UNIQUE(nom, prenom) );
```

```
ALTER TABLE personne
MODIFY(nom VARCHAR(50));
```

```
ALTER TABLE personne
DROP CONSTRAINT pos_num_h;
```

DROP TABLE

- **Utilisation**

  Suppression d'une relation (si l'utilisateur dispose de tels droits)

- **Intégrité référentielle**

  En cas de problème d'intégrité référentielle, le mot clé CASCADE CONSTRAINTS permet de supprimer en cascade toutes les associations pointant sur la clé primaire de la table à détruire.

- **Syntaxe**

  DROP TABLE <nom table> [CASCADE CONSTRAINTS]

Création de vues : CREATE VIEW

- **Principe**
  - définition comme résultat d'une requête de sélection SQL sur le contenu de la BD
  - utilisation (parmi d'autres) pour mémoriser un résultat temporaire
- **Syntaxe**

  CREATE VIEW <nom_vue> AS <requête SQL> ;
- **Exemples**

```
CREATE VIEW nom_personnes
AS SELECT nom, prenom FROM personne ;
```

```
CREATE VIEW nb_pers_parti
AS     SELECT parti, COUNT(num_h)
       FROM personne
       GROUP BY parti ;
```

- **Suppression** DROP VIEW <nom_vue>

Bases de données

SQL et LMD

Langage de Manipulation de Données

L3 S&T Mention Informatique

Insertion de tuples

- **Insertion sous l'interface conversationnelle SQL**
  - Insertion par sélection de valeurs dans la base de données ou par définition directe des valeurs de ces nouvelles occurrences
  - Insertion renseignant tous les attributs ou seulement certains attributs

- **Syntaxe** INSERT INTO <table [(<liste attributs>)] <requête SQL>
  INSERT INTO <table [(<liste attributs>)] VALUES (<liste valeurs>)

- **Exemples**

```
INSERT INTO personne VALUES (12, 'Sarkozy ', 'Nicolas ',6);
INSERT INTO personne(num_h,nom,prenom)
        VALUES (13, 'Raffarin ', 'Nicolas ');
INSERT INTO personne SELECT * FROM personnes_UMP;
```

- **Insertion par importation de données** SQL*Loader

Mise à jour de tuples

- **Syntaxe**

```
UPDATE <table>
SET <attribut> = <valeur>

{ WHERE <condition> } ;
```

⇨ *table concernée*

⇨ *valeur modifiée de l'attribut modifié (issue le cas échéant de l 'exécution d'une requête SQL).*

⇨ *sélection des tuples concernés par la modification*

- **Exemples**

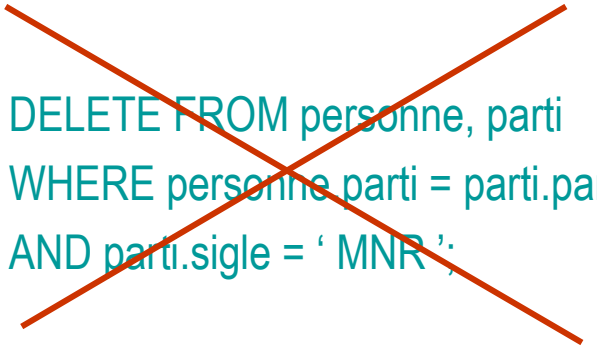
```
UPDATE personne SET parti = 6 WHERE parti = 4;

UPDATE personne
SET parti = ( SELECT parti FROM personne
                WHERE nom=‘ Chirac ’
                        )
WHERE nom=‘ Debré ’;
```

Suppression de tuples

- **Principe** on sélectionne les tuples à supprimer suivant un critère donné
- **Syntaxe** DELETE FROM <table> WHERE <condition>
- **Exemples** DELETE FROM personne WHERE nom=‘ Jospin ’;

```
DELETE FROM personne
WHERE parti IN (    SELECT parti_id FROM parti
                    WHERE sigle = ‘MNR ’) ;
```



```
DELETE FROM personne, parti
WHERE personne.parti = parti.parti_id
AND parti.sigle = ‘ MNR ’;
```