SGBD - 2e

Chapitre 3 - Langage de définition des données (LDD)

Daniel Schreurs

10 août 2022

Haute École de la Province de Liège

Table des matières du chapitre i

Introduction

Table des matières de la section : Introduction i

Table des matières de la section : Introduction ii

Introduction : Objectifs

Dans ce chapitre, nous allons étudier les commandes principales pour :

- définir,
- modifier,
- supprimer

un objet d'une base de données relationnelle

Introduction : BNF

- La forme de Backus-Naur (BNF : Backus-Naur Form) est une notation permettant de décrire les règles syntaxiques des langages de programmation.
- Elle est utilisée dans certains livres pour décrire le langage étudié, également par de nombreux logiciels d'analyse syntaxique.
- C'est une notation pour des grammaires formelles de type hors-contexte.

Introduction : BNF

Symbole	Signification
::=	Définit comme suit
|	”ou” logique
<>	Concept (nom d’objet, valeur, …)
[ ]	Option possible, non obligatoire
{ }	Élément à choisir, liste d’éléments à choisir

Introduction : Avertissement

Remarque très importante
Au cours théorique, nous voyons la norme SQL2. Au laboratoire, nous mettons en pratique dans l'environnement Oracle. Oracle n'implémente pas l'entièreté de la norme, en particulier, l'objet Domaine n'existe pas en Oracle !

Domaines

Table des matières de la section : Domaines i

Table des matières de la section : Domaines ii

Domaines : Création

CREATE DOMAIN

```
CREATE DOMAIN nom type [valeur] ;
valeur ::= DEFAULT constante
  |  USER
  |  NULL
  |  CURRENT_DATE
  |  CURRENT_TIME
  |  CURRENT_TIMESTAMP
type ::= CHAR [ (n) ]  |  VARCHAR [ (n) ]
  |  SMALLINT  |  INTEGER
  |  NUMERIC [ (p [, q]) ]
  |  DECIMAL [ (p [, q]) ]
  |  FLOAT  [ (n) ]
  |  DATE [ ANSI | VMS ]
  |  TIME frac
  |  TIMESTAMP frac
  |  INTERVAL type_intervalle
```

Domaines : Valeurs possibles

Les valeurs possibles

- *Constante* : un nombre, une chaîne de caractères ou une date
- *USER* : nom de l'utilisateur (user name) du processus qui a lancé une session SQL interactive ou qui a lancé l'exécution d'un programme
- *NULL* : valeur indéfinie des bases de données
- *CURRENT_DATE* : la date du jour (année, mois, jour)
- *CURRENT_TIME* : le temps courant (heure, minute, seconde)
- *CURRENT_TiMESTAMP* : la date et le temps courants (jour, mois, année, heure, minute, seconde)

Domaines : CHAR et VARCHAR

Différence entre CHAR et VARCHAR :

- CHAR(40) : la taille du champ aura toujours la longueur 40, la longueur est fixe
- VARCHAR(40) : le 40 représente la taille maximale permise pour le champ, si on ne met que 1 caractère, on n'utilisera que la place nécessaire pour 1 caractère.

Domaines : CHAR et VARCHAR

Types :

- SMALLINT : 2 bytes
- INTEGER : 4 byes
- NUMERIC/DECIMAL : la différence est la représentation de la donnée en mémoire
- DECIMAL : (nombre de chiffres dans le nombre + 1) / 2 $\Longrightarrow$ nb bytes nécessaires
- NUMERIC : 1 byte par chiffre

Domaines : Type DATE

SQL2 a apporté une extension au type DATE.

- Le type datetime
  - DATE (ex. 1957-01-11)
  - TIME (ex. 21:30:15.54)
  - TIMESTAMP (ex. 1957-01-11:21:30:15.54)
- Le type interval [2].
  - Année-mois
  - Jour-temps

2. peut être négatif

Domaines : Type DATE

- Les intervalles Année-Mois comprennent :
  - YEAR
  - YEAR TO MONTH
  - MONTH
- Les intervalles Jour-Temps comprennent :
  - DAY
  - DAY TO HOUR
  - DAY TO MINUTE
  - DAY TO SECOND
  - HOUR
  - HOUR TO MINUTE
  - HOUR TO SECOND
  - MINUTE
  - MINUTE TO SECOND
  - SECOND

Domaines : Type DATE

Exemple : CREATE DOMAIN

```
1 CREATE DOMAIN NomAuteur CHAR (20);
2 CREATE DOMAIN StatutEmploye SMALLINT DEFAULT 10;
3 CREATE DOMAIN AdresseDeuxiemeResidence VARCHAR (70)
4   DEFAULT NULL;
5 CREATE DOMAIN Option CHAR (15) DEFAULT 'informatique';
6 CREATE DOMAIN DateNaissance DATE;
7 CREATE DOMAIN TempsPreste INTERVAL DAY TO HOUR;
```

Relations

Table des matières de la section : Relations i

Table des matières de la section : Relations ii

Relations : CREATE TABLE

CREATE TABLE

```
CREATE TABLE nom_table
 (liste_def_colonne [liste_contrainte_table]);

def_colonne ::= nom_colonne    type | nom_domaine
 [val_par_defaut]  { contrainte_colonne }

contrainte_colonne ::= CONSTRAINT nom_contrainte
 type_contrainte_col mode_contrainte

type_contrainte_col ::= PRIMARY KEY
 | NOT NULL
 | UNIQUE
 | CHECK ( condition )
 | REFERENCES nom_table ( liste_colonne )
          [ON DELETE { NO ACTION | CASCADE | SET
              DEFAULT | SET NULL}]
          [ON UPDATE { NO ACTION | CASCADE | SET
```

Relations : CREATE TABLE

- Dans un même schéma, le nom des tables doit être unique
- Avec oracle : un type par colonnes [3].
- On peut avoir plusieurs contraintes sur une même colonne
- On peut préciser des contraintes au niveau de la colonne en cours
- *CHECK* permet de mettre des contraintes applicatives. [4]
- Pour les clés étrangères : ON DELETE et ON UPDATE.

3. Puisque la notion de domaine n'existe pas en Oracle
4. On peut mettre une condition composée avec des ET et des OU.

Relations : CREATE CONSTRAINT

CREATE CONSTRAINT

```
contrainte_table ::= CONSTRAINT nom_contrainte
 type_contrainte_table mode_contrainte

type_contrainte_table ::=
      PRIMARY KEY ( liste_colonne )
     | UNIQUE ( liste_colonne )
     | CHECK ( condition )
     | FOREIGN KEY ( liste_colonne )
            REFERENCES nom_table ( liste_colonne )
               [ON DELETE { NO ACTION | CASCADE | SET
                   DEFAULT | SET NULL}]
               [ON UPDATE { NO ACTION | CASCADE | SET
                   DEFAULT | SET NULL}]

mode_contrainte ::= [  NOT ] DEFERRABLE
```


5. Le mode de la contrainte est différé ou non. Par défaut, une contrainte n'est pas

Relations : CREATE CONSTRAINT

- Par défaut, une colonne est ”nullable” (NULL)
- L’ensemble formé par les noms des colonnes doit être unique au sein de la table
- L’ensemble formé par le nom des contraintes doit être unique au sein du schéma

Relations : CREATE CONSTRAINT

- La casse, n’a pas d’importance au niveau des mots-clés de SQL, des noms de tables, colonnes, index, etc.
- La casse a une incidence majeure dans les expressions de comparaison entre colonnes et valeurs

Relations : Formes de contraintes

Deux formes de contraintes :

- Une contrainte de colonne permet d'associer une contrainte à UNE colonne
- Une contrainte de table permet de définir une contrainte au niveau de la table, et donc faire intervenir plusieurs colonnes :
  - Clé primaire ou étrangère composée de plusieurs colonnes
  - Comparaison des valeurs contenues dans 2 colonnes (date de naissance $\leq$ date décès)

Relations : Types de contraintes

Plusieurs types de contraintes :

- Sur les valeurs directement : DEFAULT / NOT NULL / UNIQUE
- Sur les intégrités : PRIMARY KEY, FOREIGN KEY, SK (secondary key)
- Sur des conditions à remplir : CHECK

Relations : Bonnes pratiques

Sans la clause CONSTRAINT, SQL génère automatiquement un nom pour chaque contrainte. Il est donc vivement conseillé d'utiliser systématiquement la clause CONSTRAINT

Exemple : nom de contrainte

```
CREATE TABLE medecins (
     NrMedecin CHAR(11)
         CONSTRAINT CPMedecins PRIMARY KEY,
     Nom VARCHAR2(20)
         CONSTRAINT MedecinsNomNN CHECK (Nom IS NOT NULL
             )
         CONSTRAINT MedecinsNomLg CHECK (length(Nom) >=
             2),
);
```

Relations : Contraintes sur les valeurs propres

- Principe : restreindre les conditions d'acceptation des données dans la ou les colonnes
- 3 contraintes de valeurs propres : NOT NULL, DEFAULT, CHECK

Exemple : nom de contrainte

```
<contrainte de valeur> ::=
     [CONSTRAINT nom_contrainte]
     {[NOT] NULL | DEFAULT <expression_defaut>
         | CHECK (expression_validation)}
```

Relations : Contraintes sur les valeurs propres

- Il ne faut pas confondre NULL qui est une ABSENCE de valeur ou un ensemble vide avec :
  - Un numérique dont la valeur est zéro (qui est une valeur)
  - Une chaîne de caractères vide (qui a un sens)

NULL est un marqueur

Relations : Contraintes sur les valeurs propres

Valeurs par défaut d'une colonne (expression_defaut)

- Constantes les plus courantes : *CURRENT_DATE*, *CURRENT_TIME*, *CURRENT_TIMESTAMP* etc.
- Toute valeur scalaire définie par l'utilisateur
- Remarque : si une colonne a une valeur par défaut et est définie par un domaine ayant une valeur par défaut, la première prend le pas sur la seconde.

Relations : Contraintes sur les valeurs propres

Contrainte de validation (CHECK)

- Permet de restreindre les valeurs acceptables pour la colonne en appliquant un PREDICAT.
- Si le prédicat est VRAI, l'instruction est acceptée et la valeur de la colonne du tuple est validée
- Ce type de contrainte peut viser à valider plusieurs colonnes simultanément (contrainte de table) ou une seule colonne (contrainte de colonne)

Relations : Contraintes sur les valeurs propres

Contrainte de validation (CHECK) peut contenir :

- Des valeurs explicites
- Le marqueur NULL
- Des valeurs sous forme de fonction SQL ou UDF
- Des opérateurs algébriques $+, -, *, /$
- Des concaténations de chaînes $||$
- Des opérateurs de comparaisons $>, <, <=, >=, <>$
- Des connecteurs logiques *AND*, *OR*
- L'opérateur de négation *NOT*
- La hiérarchisation des opérateurs à l'aide de parenthèses
- Des expressions SQL spécifiques, y compris d'appel à d'autres tables

Relations : Contraintes sur les valeurs propres

Opérateurs SQL

Opérateur	Signification
BETWEEN	Plage de valeurs (bornes comprises dans l'intervalle)
LIKE	Comparaison partielle de chaines de caractères (%, _)
IN	Liste de valeurs possibles
CASE	Branchement de différentes valeurs
OVERLAPS	Recouvrement de périodes
SIMILAR	Expression rationnelle

Relations : Contraintes d'intégrités

Contraintes portant sur les intégrités [6]

- Permettent de restreindre les valeurs des données de différentes colonnes par rapport à un ensemble de données pris dans l'intégralité des lignes d'une table.
- 3 types de contraintes de clé :
  - Primary key
  - Foreign key
  - Unique

6. Ces contraintes ne peuvent être bâties sur une colonne de type LOB (CLOB/BLOB/NCLOB) ou de type objet.

Relations : Contraintes d'intégrités

Contrainte de colonne

```
<contrainte de clef> ::=
  [CONSTRAINT nom_contrainte]
  {PRIMARY KEY
     | UNIQUE
     | REFERENCES <table_mere> (colonne de ref)}
```

Contrainte de table

```
<contrainte de clef> ::=
  [CONSTRAINT nom_contrainte]
  {PRIMARY KEY (<liste_colonne>)
     | UNIQUE (<liste_colonne>)
     | FOREIGN KEY  (<liste_colonnes>) REFERENCES <
        table_mere> (liste de colonnes de ref)}
```

Relations : Clé primaire

Important
Il ne peut y avoir qu'une seule contrainte de clé primaire par table

- Les colonnes participant à la clé primaire ne peuvent pas être marquée "NULLable"
- Les colonnes faisant partie de la définition d'une contrainte de clé primaire seront automatiquement basculées à NOT NULL

Relations : Clé subrogée (UNIQUE)

- Impose que les occurrences de toutes les valeurs renseignées pour la ou les colonnes faisant partie de cette clé soient différentes (sans doublons)
- Marqueur NULL peut être présent plusieurs fois ;
- Autant de contraintes UNIQUE que l'on veut
- La contrainte UNIQUE n'impose pas le NOT NULL

Relations : Clé étrangère

- Permet de lier les tables du modèle relationnel
- Permet de vérifier la présence des valeurs dans les colonnes d'une autre contrainte (primaire OU subrogée) prise dans une autre table [7]
- Lié à une clé primaire ou subrogée

7. Dite table mère

Relations : Clé étrangère

Contrainte FK

```
<contrainte de clef> ::=
[CONSTRAINT nom_contrainte]
 FOREIGN KEY (<liste_colonnes>)
 REFERENCES <table_mere> (liste de colonnes
                                de ref)
 [<clause de validation>]
 [<clause de gestion-modification>]
 [<clause de gestion-suppression>]
```

Relations : Clé étrangère

- Pour insérer ou modifier une valeur de colonne pourvue d'une clé étrangère, la valeur (référençante) doit exister dans la table mère (valeur référencée)
- Types des colonnes : domaines compatibles
- Les noms des colonnes référençantes et référencées peuvent être différents. [8]

8. Mais il est conseillé de leur donner le même nom car elles sont sémantiquement liées. Cela permet également les jointures naturelles.

Relations : Clé étrangère

Exemple contrainte 1

```
CREATE TABLE Auteurs
( NumAuteur NumeroAuteur
 CONSTRAINT CPAuteur PRIMARY KEY NOT DEFERRABLE,
  Nom  NomAuteur,
  Prenom PrenomAuteur);
```

Deux manières équivalentes

Exemple contrainte 2

```
CREATE TABLE Auteurs
( NumAuteur NumeroAuteur,
  Nom  NomAuteur,
  Prenom PrenomAuteur,
  CONSTRAINT CPAuteur
 PRIMARY KEY (NumAuteur) NOT DEFERRABLE);
```

Relations : Tables temporaires

Il est également possible de créer des tables temporaires :

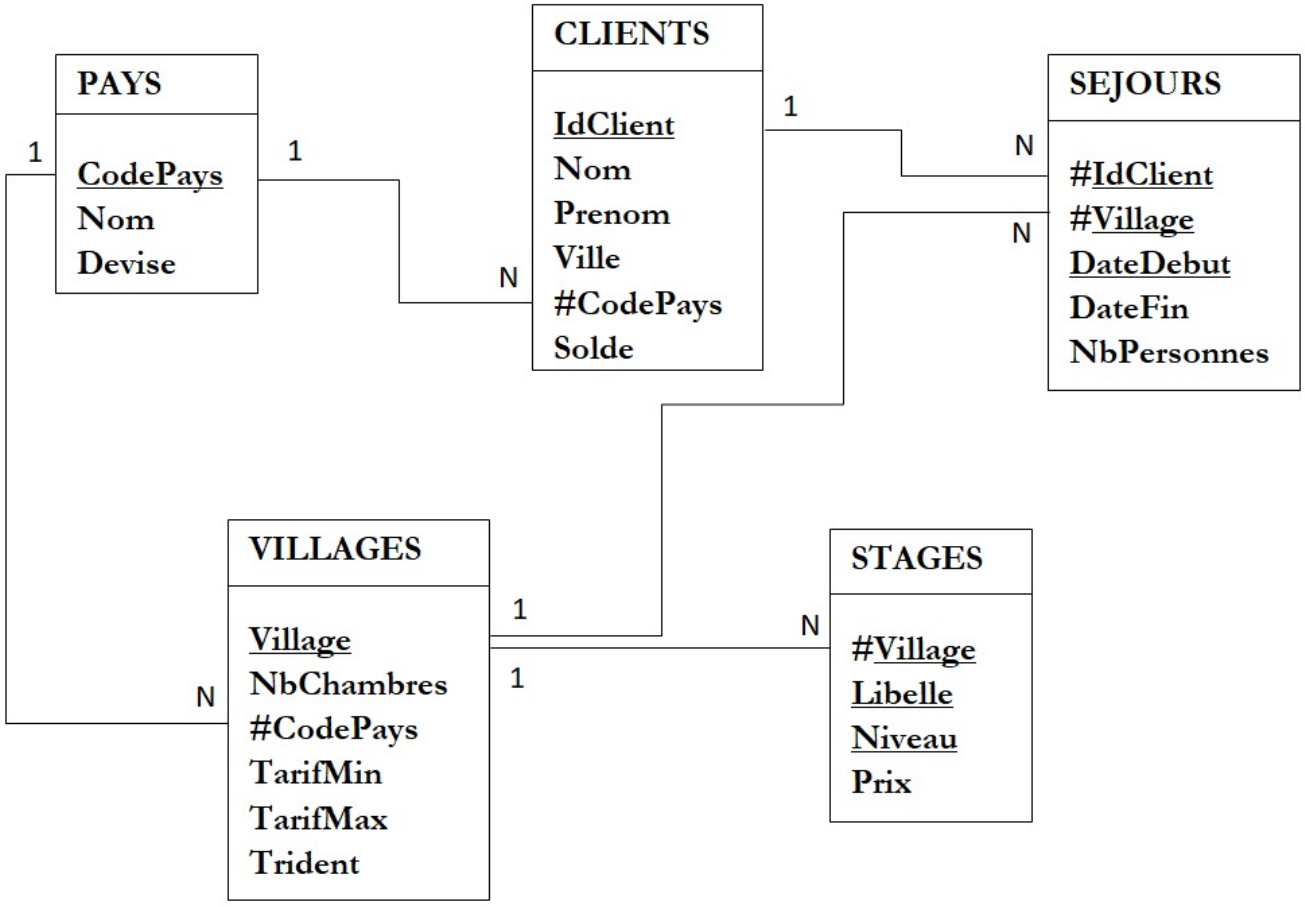
Exemple contrainte 2

```
CREATE TEMPORARY TABLE NomTable
[ON COMMIT {PRESERVE | DELETE} ROWS ];
```

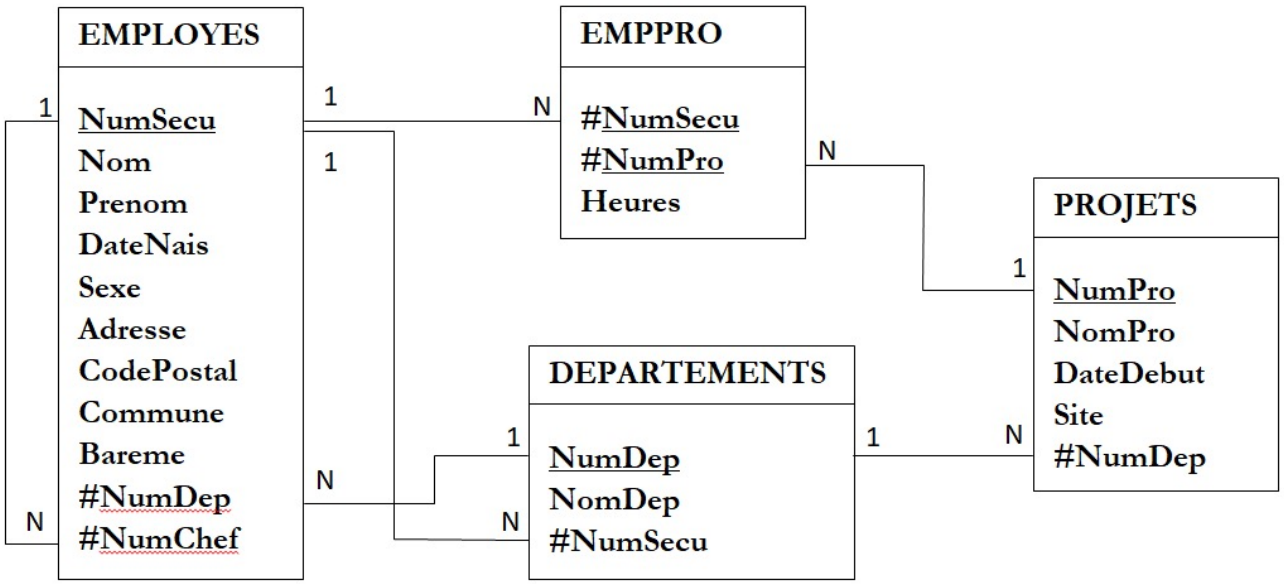
Ces tables :

- Sont vides au début de chaque session SQL,
- Peuvent être vidées à chaque COMMIT,
- Sont supprimées automatiquement à la fin de la session SQL
- Sont instanciées lors de la première référence (chaque session possède sa propre version, donc pas d'accès concurrents !)

Relations : Exercice



Relations : Exercice



Bases de données

Table des matières de la section : Bases de données i

Table des matières de la section : Bases de données ii

Bases de données : Généralités

- La norme SQL2 ne définit pas formellement la notion de *base de données* ;
- Elle indique que toutes les opérations SQL doivent être exécutées au sein d'un environnement appelé SQL-environment ;
- Un environnement peut contenir plusieurs catalogues. Chaque catalogue est constitué d'un ensemble de schémas ;
- Un schéma est une collection de domaines, de relations, de contraintes, de vues et de privilèges [9] ;
- La norme ne définit pas de mécanismes explicites de création et de destruction des catalogues.

9. Notion la plus proche d'une base de données

CREATE SCHEMA

```
CREATE SCHEMA
[NomSchema] [AUTORIZATION nom]
[DEFAULT CHARACTER SET JeuCaracteres]
[ListeElementSchema];
```

ListeElementsSchema permet de définir les domaines, les relations, les vues, les contraintes et les autorisations d'accès contenus dans le schéma. Il est possible d'effectuer des opérations entre objets de schémas différents : *Test*.*Bibliotheque*.*Membres* désigne la table Membres du schéma Bibliothèque du catalogue Test

Bases de données : Généralités - Oracle

- Une base de données Oracle est constituée d'une collection de schémas (users).
- Un schéma contient des objets du monde relationnel : table, vue, index, ainsi que des déclencheurs, fonctions, procédures stockées et packages...
- Ces schémas sont répartis dans des zones logiques appelées tablespaces constituées de segments (segments) composés de différentes extension (extents) organisées en blocs (blocks).

Aperçu des instructions Oracle pour créer une base de données, un tablespace,...

Quelques instructions Oracle

```
CREATE DATABASE NomBase;
CREATE TABLESPACE NomTableSpace;
CREATE ROLLBACK SEGMENT NomSegment;
CREATE USER sgbdTest IDENTIFIED BY sgbdTest
  DEFAULT TABLESPACE users
  TEMPORARY TABLESPACE temp
  PROFILE DEFAULT ACCOUNT UNLOCK;
```

Bases de données : Information schema

- Chaque catalogue contient un schéma spécial, *Information schema* qui contient la définition de tous les objets inclus dans les différentes schémas du catalogue.
- Ce schéma est appelé *dictionnaire* ou *méta-base*
- Il comporte un ensemble de tables, vues et contraintes créées et maintenues automatiquement par le SGBD.
- Il est accessible (pour consultation uniquement) aux utilisateurs de la base au moyen de SQL.

Bases de données : Information schema

Principales tables et vues contenues dans le dictionnaire :

- Informations générales sur la base
- Domaines et colonnes
- Tables et vues
- Droit d’accès
- Contraintes

Bases de données : Information schema

Nom de la table	Informations
INFORMATION_SCHEMA_CATALOG_NAME	Table d'une ligne et une colonne qui contient le nom du catalogue dans lequel le information schema réside.
SCHEMATA	Contient la liste des schémas créés par l'utilisateur courant.

Informations générales sur la base

Bases de données : Information schema

DOMAINS	Contient la liste des domaines
COLUMNS	Contient la liste des colonnes des tables et des vues
COLUMN_DOMAIN_USAGE	Contient le domaine de définition de chaque colonne

Domaines et colonnes

Bases de données : Information schema

TABLES	contient le liste des tables et vues
VIEWS	contient la définition des vues
VIEW_TABLE_USAGE	indique les tables (ou les vues) dont dépend chaque vue
VIEW_COLUMN_USAGE	indique les colonnes des tables (ou des vues) dont dépend chaque vue

Tables et vues

Bases de données : Information schema

TABLE_PRIVILEGES	liste des privilèges d'accès aux tables ou aux vues
COLUMN_PRIVILEGES	liste des privilèges d'accès aux colonnes
USAGE_PRIVILEGES	liste des privilèges d'accès aux domaines

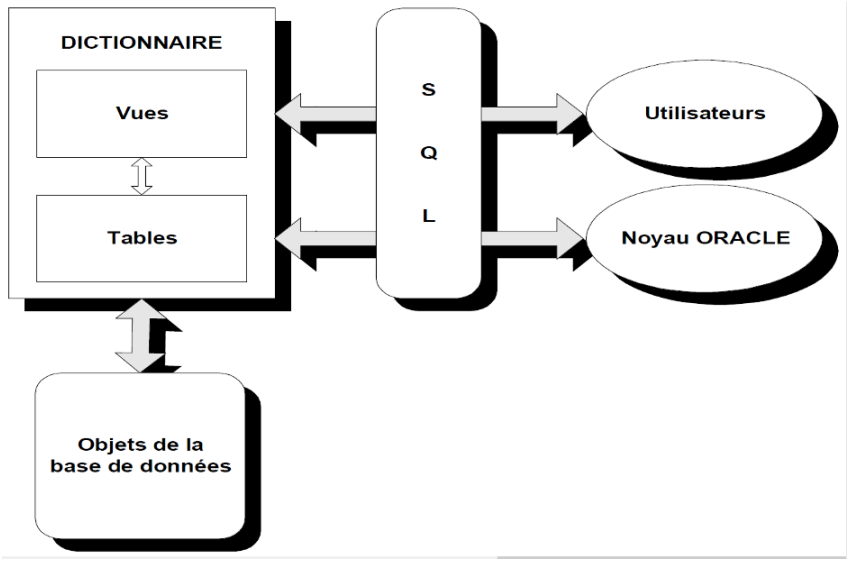
Droit d'accès

Bases de données : Information schema

DOMAIN_CONSTRAINTS	liste des contraintes portant sur les domaines
TABLE_CONSTRAINTS	liste des contraintes portant sur les tables
REFERENTIAL_CONSTRAINTS	liste des contraintes référentielles
CHECK_CONSTRAINTS	liste des contraintes de type CHECK
KEY_COLUMN_USAGE	liste des colonnes participant à une clé candidate ou à une clé étrangère
ASSERTIONS	liste des contraintes générales
CONSTRAINT_TABLE_USAGE	liste des tables intervenant dans une contrainte
CONSTRAINT_TABLE_USAGE	liste des colonnes intervenant dans une contrainte

Contraintes

Bases de données : Information schema - Oracle



Structure et utilisation du dictionnaire de données en Oracle

Bases de données : Information schema - Oracle

- Les vues relatives aux objets d'un utilisateur :
  - *USER_TABLES*
  - *USER_OBJECTS*
  - ...
- Les vues relatives aux objets accessibles à un utilisateur :
  - *ALL_TABLES*
  - *ALL_OBJECTS*
  - ...
- Les vues relatives aux administrateurs :
  - *DBA_DATA_FILES*
  - *DBA_CATALOG*
- Les vues relatives au suivi des performances :
  - *V$LOGFILE*
  - *V$SESSION*

Index

Table des matières de la section : Index i

Table des matières de la section : Index ii

Index : La méthode séquentielle

La méthode séquentielle est très simple et ne nécessite aucune gestion supplémentaire pour le SGBD. Elle convient bien pour :

- Les parcours complets
- Les opérations d'ajouts massifs
- Elle devient vite inefficace pour résoudre des requêtes contenant des jointures ou n'extrayant que peu de tuples d'une table.

Index : La méthode séquentielle

Important

Augmenter la rapidité d'accès aux tuples en proposant un accès plus direct que l'accès séquentiel. [10]

- Un index peut être défini sur plusieurs colonnes ? **V**/F
- Un index ne porte que sur une seule relation ? V/**F**
- Une relation peut posséder plusieurs index ? **V**/F [11]

10. Dès qu'on définit la clé primaire, EN ORACLE (pas dans la norme), un index est créé sur cette clé primaire.
11. Attention, lors des mises à jour, les index doivent aussi être mis à jour $\Longrightarrow$ plus lent.

Index : Index hiérarchiques

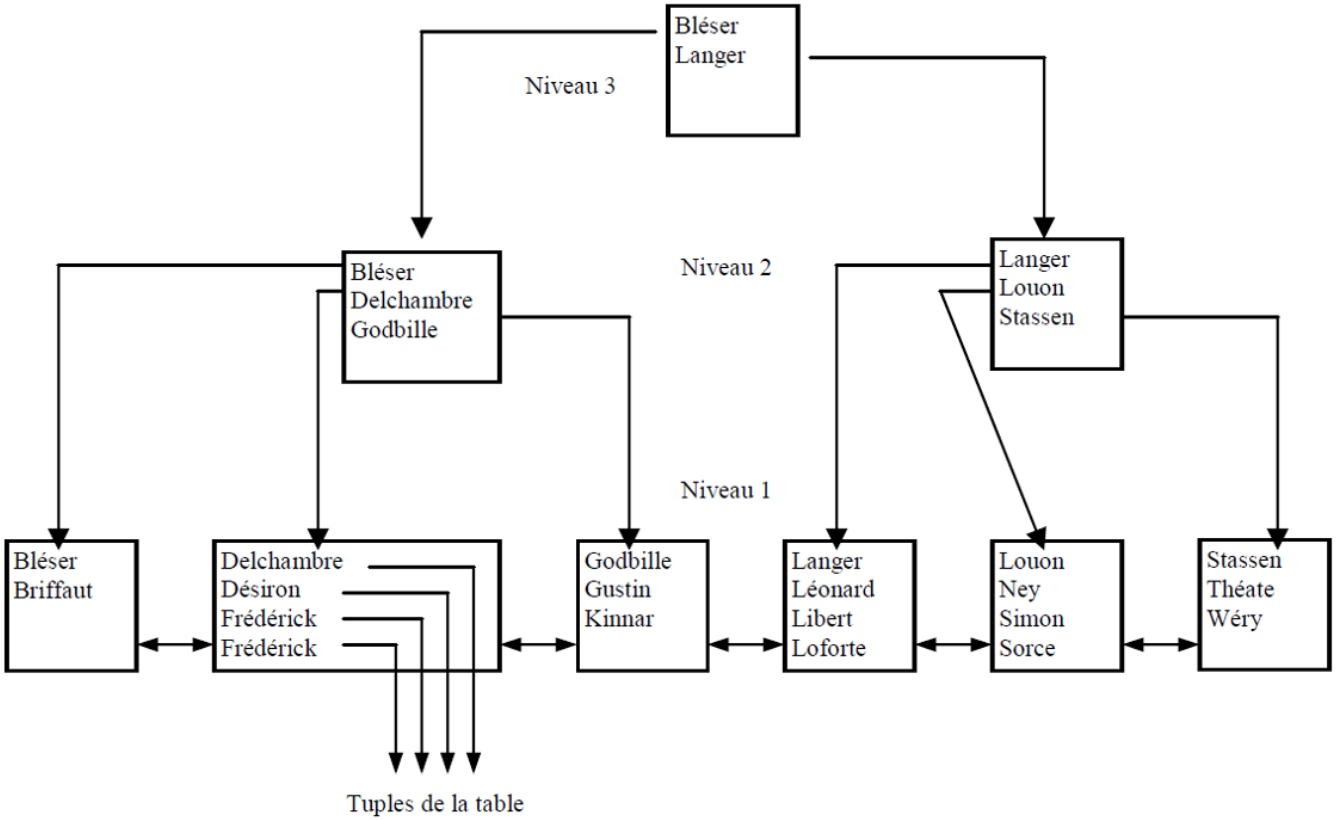
Un index peut être défini sur une seule relation, à partir d'une ou plusieurs colonnes.

CREATE INDEX

```
CREATE [UNIQUE] INDEX nom_index ON nom_table
     ( liste_colonne_tri );
colonne_tri ::= nom_colonne [ ASC | DESC ]
```

- Il est possible de définir plusieurs index sur une même relation, mais ce n'est pas toujours souhaitable.
- Un index hiérarchique ressemble à un arbre dont les feuilles pousseraient vers le bas et peut être comparé à un fichier séquentiel indexé.
- Pratiquement tous les SGBD (dont Oracle) utilisent les index hérarchiques B-TREE.

Index : Index hiérarchiques



Index hiérarchiques

Index : Index hiérarchiques

Résumé des principales caractéristiques

- Offrent des recherches rapides en fonction d'une valeur précise de la clé ou d'une plage de valeurs pour la clé
- conviennent pour les colonnes qui ont un grand nombre de valeurs différentes et qui interviennent dans les critères de recherches ou de jointures
- en pratique, on en définit toujours pour les clés primaires et les clés étrangères

Index : Cluster indexé

Un cluster contient les tuples d'une ou plusieurs tables qui possèdent au moins une colonne commune. Les tuples qui partagent la même valeur pour ces colonnes communes sont physiquement stockés ensemble dans la base de données.

Important

Ce type de structure optimise le temps nécessaire aux opérations de jointure.

Cluster indexé

```
1 CREATE CLUSTER membres_et_emprunts
2     (num_membre char (4));
```

Index : Cluster indexé

Cluster indexé

```
1  CREATE TABLE membres
2  (
3      num_membre char(4)
4          CONSTRAINT CPMembres PRIMARY KEY,
5      nom         varchar2(32),
6      prenom      varchar2(32)
7  ) CLUSTER membres_et_emprunts (num_membre);
8
9  CREATE TABLE emprunts
10 (
11     num_membre   char(4)
12         CONSTRAINT RefNumMembreMembres
13             references Membres (num_membre),
14     num_ouvrage  char(6)
15         CONSTRAINT RefNumOuvrage
16             references Ouvrages (num_ouvrage),
17     date_emprunt date
```

Index : Cluster indexé

Résumé des principales caractéristiques

- En regroupant physiquement les lignes de plusieurs tables, améliorent les temps de jointures.
- MAIS pénalisent les parcours séquentiels

Index : Index haché

- Le SGBD applique une fonction mathématique, appelée fonction de hachage, sur une clé. Le résultat est un numéro de bloc dans un fichier ou zone de stockage. Le tuple
- Le numéro de bloc dépend du nombre de blocs initialement alloués à la zone de stockage et, évidemment, de la clé.
- Il ne peut donc y avoir qu'un seul hachage par relation.
- En théorie, cette technique permet d'accéder à un tuple en une entré/sortie.

Index : Index haché

Important

Problème de collisions [12] :

- Chaînage des collisions
- Application d'une fonction secondaire

12. Les collisions augmentent le nombre d'entrées/sorties nécessaires à la recherche d'un tuple et brisent l'ordre logique.

Index : Index haché

- Pour limiter les collisions au maximum, le concepteur de la base de données doit choisir un nombre de blocs initiaux suffisamment grand. Pour une grande relation, il n'est pas rare de définir une réserve d'espace de 50
- La répartition des tuples dans les blocs sera d'autant meilleure que le nombre de clés différentes est plus élevé.
- Le hachage à partir d'une colonne contenant beaucoup de valeurs NULL est donc à proscrire.

Index : Index haché

- Exemple : le hachage est réalisé à partir de la colonne *Num_Article*. Il y aura 25000 valeurs hachées. On aura donc des collisions dès que la table Articles contiendra plus de 25000 tuples.

Cluster indexé

```
CREATE
CLUSTER klu (num_article number (5, 0)
    hash is num_article
    hashkeys 25000;

CREATE TABLE Articles (
    num_article number (5, 0)
        CONSTRAINT CPArticles PRIMARY KEY,
    nom  varchar2 (100),)
CLUSTER klu;
```

Index : Index haché

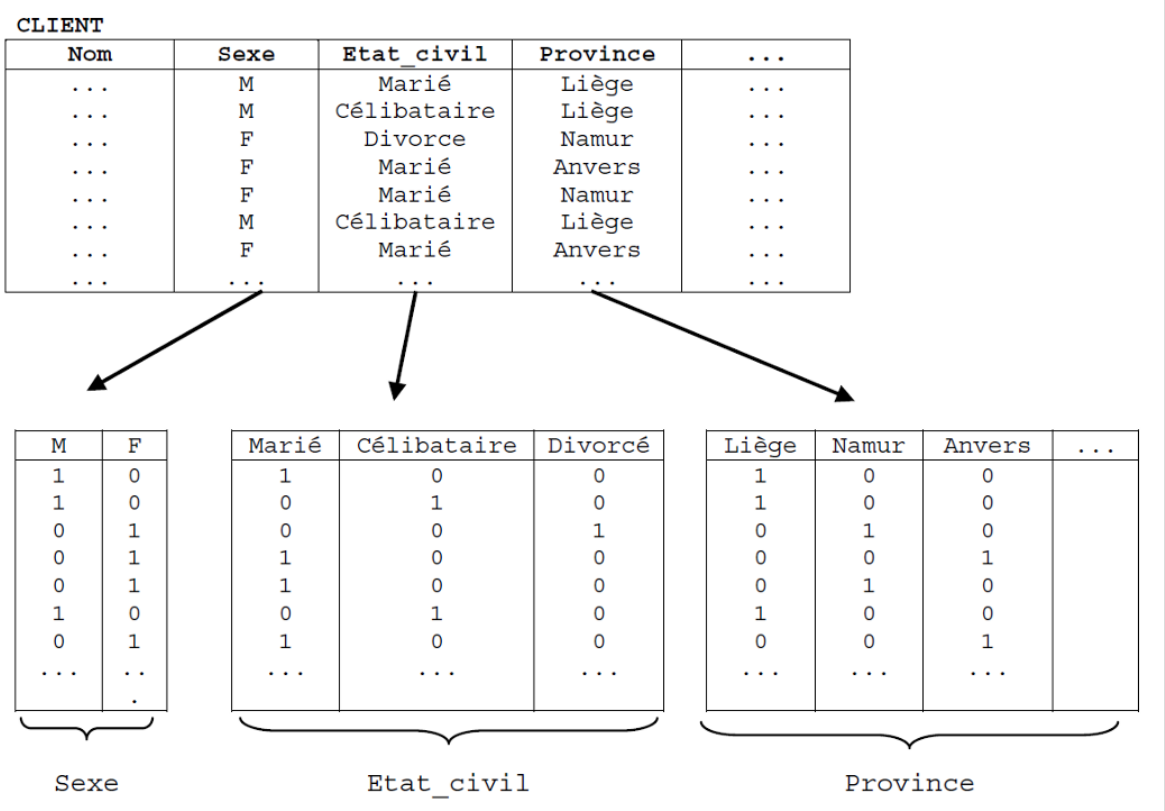
Résumé

Permettent en une seule opération d'entrée/sortie, de retrouver un tuple dont on donne la valeur de la clé.

Index : Index à matrices binaires

- Colonnes qui ne possèdent que quelques valeurs distinctes
- Un index à matrices binaires est constitué d'un ensemble de matrices dont les éléments prennent uniquement les valeurs 1 ou 0
- Une matrice est toujours associée à une des colonnes de la table indexée
- À chaque ligne de chaque matrice correspond une et une seule ligne de la table
- Le nombre de colonnes de la matrice associée à la colonne C est égal au nombre de valeurs différentes présentes dans la colonne C

Index : Index à matrices binaires



Index à matrices binaires

Index : Index à matrices binaires

Important

Ce type d'index est souvent utilisé dans le contexte des data warehouse dans lequel on manipule couramment des tables de plusieurs millions de lignes mais dont les colonnes ne contiennent que quelques valeurs différentes.

Index : Index à matrices binaires

Création d'un index bitmap en Oracle

```
1 CREATE
2 BITMAP INDEX bidx ON
3   clients (sexe, etat_civil, province);
```

Index : Index à matrices binaires

Résumé des principales caractéristiques

- sont utiles pour les requêtes de comptage en fonction de critères portant sur plusieurs colonnes contenant peu de valeurs différentes
- sont utilisés pour de grandes tables

Index : Index basé sur une fonction

À partir d'une fonction ou d'une expression faisant intervenir des colonnes de la table indexée. [13]

Création d'un index bitmap en Oracle

```
-- Avec cet index :

CREATE INDEX IdxPrixNet ON
    Articles (Prix - Ristourne);

--Oracle utilisera cet index pour ceci
SELECT *
FROM Articles
WHERE Prix - Ristourne > 200;
```

13. La fonction peut être une fonction SQL, PL/SQL ou externe.

Suppression d’un objet

Table des matières de la section : Suppression d'un objet i

Table des matières de la section : Suppression d'un objet ii

Suppression d’un objet : DROP DOMAIN

DROP DOMAIN

```
DROP SCHEMA|USER nom_schema {RESTRICT | CASCADE}
```

- RESTRICT : interdit la suppression
  - S’il y a des objets contenus
  - S’il est référencé par un autre schéma ou routine
- CASCADE : supprime
  - Tous les objets contenus
  - Tous les objets externes au schéma mais y faisant référence

Suppression d’un objet : DROP DOMAIN

- Il est impossible de supprimer un domaine qui est encore utilisé dans la définition d’une table
- Il est impossible de supprimer un index qui est utilisé en même temps dans une autre requête
- DROP DATABASE efface physiquement tout ce qui concerne la base de donnée

Suppression d'un objet : DROP DOMAIN

- DROP TABLE efface la table : ses données ET sa définition
- Si on utilise l'option RESTRICT, il est impossible de supprimer la table si :
  - La table est utilisée en même temps dans une autre requête (une sélection par ex.)
  - La table est utilisée dans la construction d'une vue
  - La table est référencée par une autre table (contrainte de référence)
- Si on utilise la clause CASCADE, l'effacement de la table entraîne automatiquement la suppression de tous les objets qui lui faisaient référence (index, vue, contrainte, déclencheur)

Modification de la définition d’un objet

Table des matières de la section : Modification de la définition d’un objet i

Table des matières de la section : Modification de la définition d'un objet ii

Modification de la définition d'un objet : Modification de la définition d'un domaine

- Syntaxe identique à celle de CREATE DOMAIN dans laquelle on remplace CREATE par ALTER
- Intérêt : d'un seul coup, on modifie la définition de toutes les relations faisant référence au domaine modifié !

Modification de la définition d'un objet : Modification de la définition d'un table

- Permet d'ajouter de nouvelles colonnes, d'ajouter des contraintes (au niveau colonne ou table), de modifier des colonnes, de supprimer des colonnes et de supprimer des contraintes
- Ne permet pas de modifier une contrainte : il faut d'abord la supprimer puis la recréer.

Modification de la définition d'un objet : Modification de la définition d'un table

Modification de la définition d'un table

```
modifier_structure_table ::=
    ALTER TABLE nom_table
           ajouter_def
        |  modifier_def
        |  supprimer_def
        ;

    ajouter_def ::= ADD
           COLUMN def_colonne
        |  CONSTRAINT contrainte_table

    modifier_def  ::= ALTER def_colonne

    supprimer_def  ::= DROP
           COLUMN nom_colonne
        |  CONSTRAINT nom_contrainte
```

Modification de la définition d'un objet : Modification de la définition d'un table

On ne peut pas modifier ou supprimer une colonne lorsqu'une des conditions suivantes est remplie :

- Elle est utilisée dans une vue
- Un index est basé sur la colonne
- La base de données contient une contrainte qui fait référence à la colonne
- ALTER TABLE échoue si on ajoute une contrainte qui n'est pas satisfaite