BNF101 Base de Données Relationnelles

SQL Langage de Définition des Données

Création des tables de la BDD COMMANDE

INTRODUCTION

- SQL : Structured Query Langage

- langage de gestion de bases de données relationnelles pour

définir les données (LDD) >> fait partie du programme du cours BNF101,

- interroger la base de données (Langage de requêtes)

Et manipuler les données (LMD) >> fait partie du programme du cours BNF101,

- contrôler l'accès aux données (LCD) >> hors programme du cours BNF101. Mais un exemple de script est donné dans le document BNF101_BDD_installation_Postgresql_macos_2024_2024_v2.pdf

HISTORIQUE

- 1974 : SEQUEL (Structured English Query Language) ancêtre de SQL
- 1979 : premier SGBD basé sur SQL par Relational Software Inc (rebaptisé Oracle)
- 1986 : SQL1 1ière norme ISO
- 1989 : ajout des contraintes d'intégrité de base (clé primaire et clé étrangère)
- 1992 : SQL2 2ième norme extension de SQL1 (nouveaux types et nouveaux opérateurs
- 1999 : SQL3 extension de SQL2 (introduction des types orientés objet)

Le LDD : Structure des tables

Les tables sont formées de lignes et de colonnes.

- Toutes les données d'une colonne sont du même type,
- Identificateur unique pour les colonnes d'une même table,
- 2 colonnes dans 2 tables différentes peuvent avoir le même nom,
- Nom complet d'une colonne comprend le nom complet de la table à laquelle elle appartient

exemple :

CLIENT.NumClient

Le LDD : création des tables

Le création de tables se fait à l'aide du couple de mots-clés CREATE TABLE. La syntaxe de définition simplifiée d'une table est la suivante:

```
CREATE TABLE Nom_de_la_table
(Nom_de_colonne1 Type_de_donnée,
 Nom_de_colonne2 Type_de_donnée,
        ...);
```

Le nom donné à la table doit généralement (sur la plupart des SGBD) commencer par une lettre, et le nombre de colonnes maximum par table est de 254.

Le LDD : les tables et les types

Pour chaque colonne que l'on crée, il faut préciser le type de données que le champ va contenir. Celui-ci peut être un des types suivants:
L'option *NOT NULL*, placée immédiatement après le type de donnée permet de préciser au système que la saisie de ce champ est obligatoire.

Type de donnée	Syntaxe	Description
Type alphanumérique	CHAR(n)	Chaîne de caractères de longueur fixe n (n<16383)
Type alphanumérique	VARCHAR(n) ou VARCHAR2(n)	Chaîne de caractères de n caractères maximum (n<16383)
Type numérique	NUMBER(n,[d])	Nombre de n chiffres [optionnellement d après la virgule]
Type numérique	SMALLINT	Entier signé de 16 bits (-32768 à 32757)
Type numérique	INTEGER	Entier signé de 32 bits (-2E31 à 2E31-1)
Type numérique	FLOAT	Nombre à virgule flottante
Type horaire	DATE	Date sous la forme 16/07/99
Type horaire	TIME	Heure sous la forme 12:54:24.85
Type horaire	TIMESTAMP	Date et Heure

Dans un SGBDR Oracle, on utilise le type NUMBER(n,d) pour décrire des valeurs numériques. Ce format n'est pas disponible dans le SGBDR Postgresql. Dans Postgresql, on utilise le type NUMERIC(n,d).

Le LDD : les tables et les types

Exemple :

Dans ORACLE

```
CREATE TABLE Produit
(NumPro NUMBER(3),
NomPro VARCHAR(30),
PrixUni NUMBER(5),
QteSto NUMBER(3));
```

Dans POSTGRES

```
CREATE TABLE Produit
(NumPro NUMERIC(3),
NomPro VARCHAR(30),
PrixUni NUMERIC(5),
QteSto NUMERIC(3));
```

EXERCICE

1/ Ecrire le script SQL de création de la table CLIENT sans les clés.

Cet exercice a pour but de se familiariser avec la syntaxe SQL la plus simple.

Nous ajouterons les clés dans un 2ème temps.

Les clés font partie de la famille des contraintes d'intégrité.

Le LDD : les contraintes d’intégrité

Une contrainte d'intégrité est une clause permettant de contraindre la modification de tables, faite par l'intermédiaire de requêtes d'utilisateurs, afin que les données saisies dans la base soient conformes aux données attendues.

Ces contraintes doivent être exprimées dès la création de la table grâce aux mots clés suivants:

- DEFAULT
- NOT NULL
- UNIQUE
- CHECK…

Le LDD : les contraintes d’intégrité exemple de la contrainte DEFAULT

- Le langage SQL permet de définir une valeur par défaut lorsqu'un champ de la base n'est pas renseigné grâce à la clause *DEFAULT*. Cela permet notamment de faciliter la création de tables, ainsi que de garantir qu'un champ ne sera pas vide.
- La clause *DEFAULT* doit être suivie par la valeur à affecter. Cette valeur peut être un des types suivants:
- constante numérique
- constante alphanumérique (chaîne de caractères)
- le mot clé **NULL**
- La valeur **0,**
- le mot clé **CURRENT_DATE** correspond à la date de saisie.

Le LDD : les contraintes d’intégrité
Autres exemples de contraintes

- La contrainte ***NOT NULL*** permet de spécifier qu'un champ doit être saisi, c'est-à-dire que le SGBD refusera d'insérer des tuples dont un champ comportant la clause *NOT NULL* n'est pas renseigné.

Ex : codePos NUMBER(5) NOT NULL

- Il est possible de faire un test sur un champ grâce à la clause ***CHECK()*** comportant une condition logique portant sur une valeur entre les parenthèses. Si la valeur saisie est différente de *NULL*, le SGBD va effectuer un test grâce à la condition logique.

Ex : total NUMBER(5) NOT NULL CHECK (total > 0)

- La clause (contrainte) ***UNIQUE*** permet de vérifier que la valeur saisie pour un champ n'existe pas déjà dans la table. Cela permet de garantir que toutes les valeurs d'une colonne d'une table seront différentes.

Le LDD : les contraintes d’intégrité
Nommage d’une contrainte

Il n’est pas obligatoire de nommer une contrainte.

Nous l’avons vu dans les exemples : codePos NUMBER(5) NOT NULL.

Ici la contrainte n’a pas de nom.

Dans ce cas, un nom sera donné arbitrairement par le SGBD.

Exemple : SYS_C0078209

Toutefois, le nom donné par le SGBD risque fortement de ne pas être compréhensible, et ne sera vraisemblablement pas compris lorsqu'il y aura une erreur d'intégrité. La stipulation de cette clause est donc fortement conseillée.

Le LDD : les contraintes d’intégrité
Nommage d’une contrainte

Il est possible de donner un nom à une contrainte grâce au mot clé *CONSTRAINT* suivi du nom que l'on donne à la contrainte, de sorte que le nom donné s'affiche en cas de non-respect de l'intégrité, c'est-à-dire lorsque la clause que l'on a spécifiée n'est pas validée.

Exemple :

Total NUMBER(5) CONSTRAINT C_nn_tot NOT NULL CONSTRAINT C_check_tot CHECK (Total > 0),

Le nom des contraintes C_nn_tot et C_check_tot est laissé au choix du développeur.

Si on choisit toto, ça marche aussi.

Le LDD : les contraintes d'intégrité
CLE PRIMAIRE

L'ensemble des colonnes faisant partie de la table en cours permettant de désigner de façon unique un tuple est appelé **clé primaire** et se définit grâce à la clause *PRIMARY KEY.*

1ère forme :

1 clé primaire peut être définie dans la ligne qui définit la colonne;

$\Rightarrow$ Voir schéma CONTRAINTE DANS LA LIGNE (IN LINE),

Exemple dans la table client:

numcli NUMBER(3) PRIMARY KEY,

Il est conseillé de nommer la contrainte (clé):

Exemple : numcli NUMBER(3) CONSTRAINT c_pk_numcli PRIMARY KEY,

Précision :

Le nom de la clé primaire c_pk_numcli est laissé au choix du développeur.

Si on choisit toto, ça marche aussi.

Le LDD : les contraintes d'intégrité
CLE PRIMAIRE

2ème forme :

1 clé primaire peut être définie après la définition des colonnes, dans 1 ligne dédiée :

⇒Voir schéma CONTRAINTE HORS DE LA LIGNE (OUT OF LINE),

Dans ce cas, il faut indiquer les éléments suivants :

- nom de la colonne de référence entre parenthèse,
- nommer la contrainte (clé),

Exemple dans la table client:

CONSTRAINT c_pk_numcli PRIMARY KEY(numcli),

Si la clé est sur 2 colonnes (clé composite), il faut les indiquer dans la parenthèse,

Exemple : PRIMARY KEY (colonne1, colonne2)

Remarque :

Par définition, une clé primaire est non nulle et unique.

Ainsi, la clause PRIMARY KEY intègre implicitement les clause NOT NULL et UNIQUE,

Le LDD : les contraintes d'intégrité
CLE ETRANGERE

Lorsqu'une colonne (ou plusieurs colonnes) de la table en cours de définition permet de référencer la clé primaire d'une table maître, on parle alors de **clé étrangère ou FOREIGN KEY.**

1ère forme :

1 clé étrangère peut être définie dans la ligne qui définit la colonne;

⇒Voir schéma CONTRAINTE DANS LA LIGNE (IN LINE),

Exemple : dans la table commande

numcli NUMBER(3) REFERENCES client(numcli),

Il est conseillé de nommer la clé :

Exemple : numcli NUMBER(3) CONSTRAINT c_fk_numcli REFERENCES client(numcli),

Précision :

Le nom de la clé étrangère c_fk_numcli est laissé au choix du développeur.

Si on choisit toto, ça marche aussi.

Le LDD : les contraintes d'intégrité
CLE ETRANGERE

2ème forme :

1 clé étrangère peut être définie après la définition des colonnes, dans 1 ligne dédiée :

⇒ Voir schéma CONTRAINTE HORS DE LA LIGNE (OUT OF LINE),

Dans ce cas, il faut indiquer éléments suivants :

- le nom de la colonne de référence entre parenthèses,
- la clause FOREIGN KEY
- nommer la contrainte (clé),

Exemple : CONSTRAINT c_fk_numcli FOREIGN KEY(numcli) REFERENCES client(numcli),

Si la clé est sur 2 colonnes (clé composite), il faut les indiquer dans la parenthèse,

Exemple : FOREIGN KEY (colonne1, colonne2, ...)

REFERENCES Nom_de_la_table_proprietaire_de_la_pk (colonne1,colonne2,...)

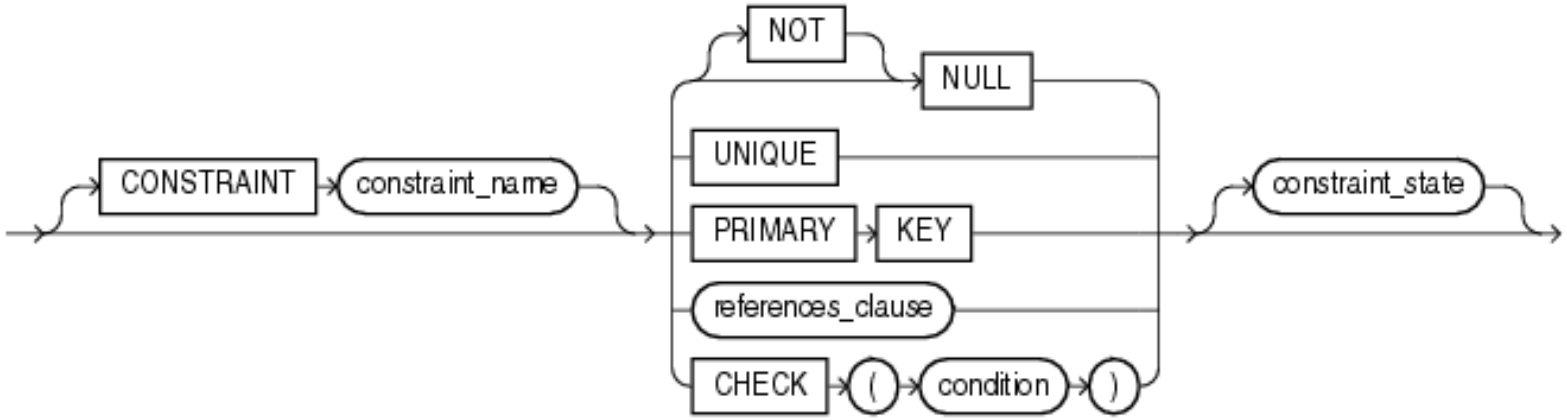
Remarque :

Par définition, une clé étrangère est non nulle.

Ainsi, la clause FOREIGN KEY intègre implicitement les clause NOT NULL,

Le LDD : les contraintes d'intégrité CONTRAINTE DANS LA LIGNE (IN LINE)

***inline_constraint*::=**



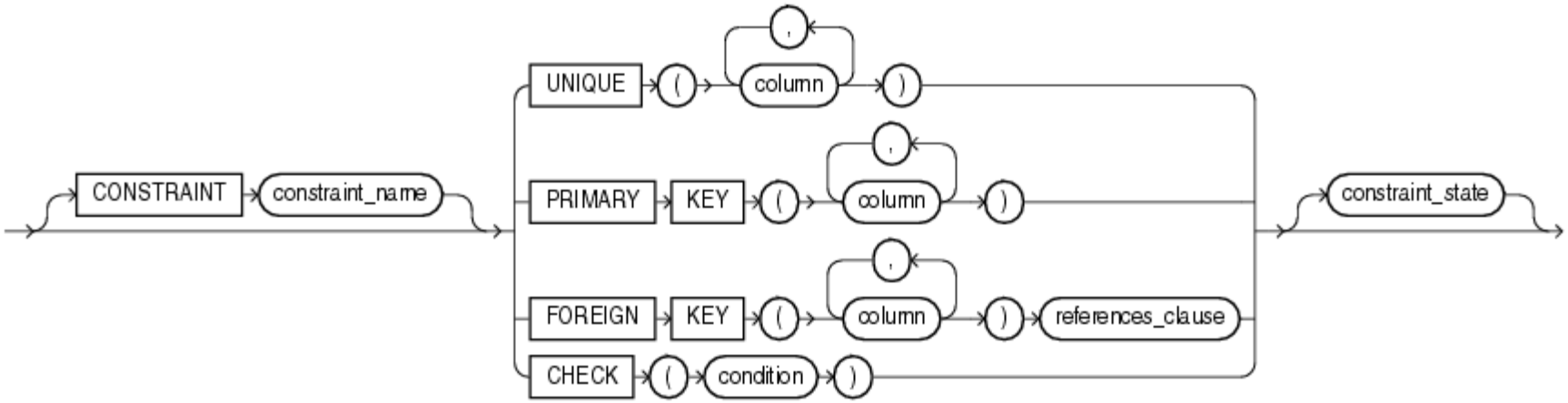
Description of the illustration inline_constraint.gif

(*references_clause*::=)

http://docs.oracle.com/cd/B19306_01/server.102/b14200/clauses002.htm

Le LDD : les contraintes d'intégrité
CONTRAINTE HORS DE LA LIGNE (OUT OF LINE)

out_of_line_constraint::=



Description of the illustration out_of_line_constraint.gif

http://docs.oracle.com/cd/B19306_01/server.102/b14200/clauses002.htm

Le LDD : Modification d’une table

Il est possible de supprimer une table grâce à la clause *DROP*, il existe aussi des commandes moins extrêmes permettant

- L'ajout de colonnes
- La modification de colonnes
- La suppression de colonnes

Il est aussi possible d'ajouter des commentaires à une table grâce à la clause *COMMENT*.

Le LDD : Modification d’une table

Suppression d’une table :

- La clause *DROP* permet d'éliminer des tables. Cette clause est toutefois à utiliser avec parcimonie dans la mesure où elle est irréversible.

La suppression d'une table se fait avec la syntaxe suivante:

DROP TABLE Nom_de_la_table;

- La clause *DROP* lorsqu'elle est utilisée sur une table élimine les données ainsi que la structure de la table. Il est possible de supprimer uniquement les données en conservant la structure de la table grâce à la clause *TRUNCATE*.

La suppression des données d'une table se fait avec la syntaxe suivante:

TRUNCATE TABLE Nom_de_la_table;

Le LDD : Modification d’une table

Renommage d’une table :

Il peut parfois être intéressant de renommer une table, c'est la clause *RENAME* qui permet cette opération. La syntaxe de cette clause sous ORACLE est :

```
alter table
   table_name
rename to
   new_table_name;
```

Le LDD : Modification d’une table

Suppression de colonnes :

La clause *ALTER* permet la modification des colonnes d'une table. Associée avec la clause *DROP COLUMN*, elle permet de supprimer des colonnes. La syntaxe est la suivante:

- ALTER TABLE Nom_de_la_table

DROP COLUMN Nom_de_la_colonne;

Il faut noter que la suppression de colonnes n'est possible que dans le cas où:

- La colonne ne fait pas partie d'une vue,
- La colonne ne fait pas partie d'un index,
- La colonne n'est pas l'objet d'une contrainte d'intégrité.

Le LDD : Modification d’une table

- Ajout de colonnes :

Associée avec la clause *ADD*, la clause *ALTER* permet l'ajout de colonnes à une table. La syntaxe est la suivante:

ALTER TABLE Nom_de_la_table

ADD Nom_de_la_colonne Type_de_donnees;

- Modification de colonnes :

Associée avec la clause *MODIFY*, la clause *ALTER* permet la modification des colonnes d’une table. La syntaxe est la suivante:

ALTER TABLE Nom_de_la_table

MODIFY Nom_de_la_colonne Type_de_donnees;

Le LDD : Modification d’une table

- Ajout de plusieurs colonnes :

ALTER TABLE table_name

ADD (column_1 column-definition, column_2 column-definition, ... column_n column_definition);

- Modification de colonnes :

ALTER TABLE table_name

MODIFY (column_1 column_type, column_2 column_type, ... column_n column_type);

Le LDD : Modification d’une table

- Ajout de contraintes :

Associée avec la clause *ADD*, la clause *ALTER* permet l'ajout de contraintes à une table. La syntaxe est la suivante:

ALTER TABLE table_name
ADD constraint constraint_name constraint_definition;

Exemple :

ALTER TABLE Client

ADD constraint c_pk_numcli PRIMARY KEY (numCli);

Le LDD : Modification d'une table

- Activation et désactivation de contraintes :

La clause *ALTER* permet l'activation ou la désactivation des contraintes.

La syntaxe est la suivante :

```
ALTER TABLE table_name
ENABLE constraint constraint_name;

ALTER TABLE table_name
DISABLE constraint constraint_name;
```

EXERCICES

2/ Sur la table CLIENT, ajouter les contraintes ou les clés au moyen de la clause ALTER.

3/ Ecrire les scripts SQL de création des tables VENDEUR, PRODUIT, COMMANDE, LIGNECOM avec les clés et en prenant en compte les contraintes suivantes :

-- table VENDEUR : les champs NomVen et Salaire ne peuvent pas être nuls.

-- table PRODUIT : le champ QteSto a comme valeur par défaut la valeur « 0 ».

-- table COMMANDE : le champ Total ne peut pas être nul et doit avoir une valeur numérique positive.

-- table LIGNECOM : le champ QteCom ne peut pas être nul et doit avoir une valeur numérique positive.

VISUALISER LES TABLES CREES SOUS ORACLE

Pour visualiser la liste des tables créées :

SQL> select table_name from user_tables;

TABLE_NAME

CLIENT

CLIENT2

LIGNECOM

CLIENTTEST

PRODUIT

COMMANDE

VENDEUR

7 rows selected.

Dans postgresql, la commande est : bnf101=# \d

VISUALISER LES TABLES CREES CREES SOUS ORACLE

Pour visualiser la structure de chaque table créée :

DESC nom_de_la_table

Remarque : le « ; » est facultatif dans cette requête.

Exemple pour la table CLIENT :

SQL> desc client

Name	Null?	Type
NUMCLI	NOT NULL	NUMBER(3)
NOMCLI		VARCHAR(30)
ADRESSECLI		VARCHAR(40)
CODEPOS		NUMBER(5)
VILLE		VARCHAR(30)
TEL		NUMBER(10)

Dans postgresql, la commande est : bnf101=# \d+ client

VISUALISER LES CONTRAINTES CREES

Pour visualiser la liste des contraintes créées dans chaque table :

select constraint_name, constraint_type from user_constraints where table_name=nom_de_la_table;

Attention la valeur du paramètre nom_de_la_table est sensible à la casse.

Exemple pour la table COMMANDE.

SQL> select constraint_name, constraint_type from user_constraints where table_name='COMMANDE';

```
 CONSTRAINT_NAME             C
------------------------------ -
SYS_C0078208                 C
SYS_C0078209                 C
C_PK_NUMCOM                  P
C_FK_NUMCLI                  R
C_FK_NUMVEN                  R
```

Exemple de requête pour visualiser les contraintes de toutes les tables :

SQL> select constraint_name, constraint_type , **table_name** from user_constraints;

VISUALISER LES CONTRAINTES CREES CREES SOUS ORACLE

Ajout de précisions dans la requête :

SQL> select constraint_name, decode(constraint_type,'P','PRIMARY KEY','C','SYSTEM','R','FOREIGN KEY') from user_constraints where table_name='COMMANDE';

CONSTRAINT_NAME DECODE(CONS
------------------------------ ----------
SYS_C0078208 SYSTEM
SYS_C0078209 SYSTEM
C_PK_NUMCOM PRIMARY KEY
C_FK_NUMCLI FOREIGN KEY
C_FK_NUMVEN FOREIGN KEY

Dans POSTGRESQL :

```
SELECT conname AS nom_contrainte,
contype AS constraint_type
 FROM pg_catalog.pg_constraint cons
 JOIN pg_catalog.pg_class t ON t.oid =
cons.conrelid
 WHERE t.relname = 'commande' ;
```

```
 nom_contrainte | constraint_type
----------------+-----------------
 c_check_tot    | c
 c_pk_numcom    | p
 c_fk_numcli    | f
 c_fk_numven    | f
(4 rows)
```

VISUALISER LES CONTRAINTES CREES CREES SOUS ORACLE

Pour voir la structure de la table système USER_CONSTRAINTS :

SQL> desc user_constraints

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(30)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)
DEFERRABLE		VARCHAR2(14)
DEFERRED		VARCHAR2(9)
VALIDATED		VARCHAR2(13)
GENERATED		VARCHAR2(14)
BAD		VARCHAR2(3)
RELY		VARCHAR2(4)
LAST_CHANGE		DATE
INDEX_OWNER		VARCHAR2(30)
INDEX_NAME		VARCHAR2(30)
INVALID		VARCHAR2(7)
VIEW_RELATED		VARCHAR2(14)