

Le Langage SQL

Compléments sur la création de tables

Le Langage de Définition de Données (DDL)

A decorative graphic on the left side of the slide, consisting of a vertical arrangement of blue and yellow hexagons of various sizes, some overlapping.

Introduction

- **Le langage de définition de données permet de créer la structure des tables qui composent votre schéma de données.**
- **Un schéma de données est issu d'un modèle conceptuel de données ou de tout autre modèle ou langage (UML par exemple) permettant la modélisation des données d'une application.**

La création de tables

■ La syntaxe générale

```
CREATE TABLE nom_table
(
  nom_colonne { type } [ DEFAULT valeur_default ][ contrainte.. ],
  .... ,
  CONSTRAINT nom_contrainte { UNIQUE |
                                PRIMARY KEY ( liste_colonne )|
                                FOREIGN KEY liste_colonne
                                REFERENCES nom_table (liste_colonne)}
);
```

La création de tables

A sa création, la table peut être remplie par une requête SELECT.
Par défaut, une table est vide à sa création.

Exemple :

-- changement de nom des colonnes sélectionnées, même type

```
CREATE TABLE Marque_Voiture  
(    fabricant,  
    Modele  
) AS SELECT MARQUE , MODELE FROM VOITURE;
```

-- même nom des colonnes sélectionnées, même type

```
CREATE TABLE Marque_Voiture_1  
AS SELECT MARQUE , MODELE FROM VOITURE;
```

-- Création de la copie d'une table

```
CREATE TABLE Save_Voiture  
AS SELECT * FROM VOITURE;
```

Le type des colonnes

- **il existe plusieurs types pour définir les colonnes d'une table :**
 - **Le type énumération**
 - **Le type tableau**
 - **Création de domaine de valeur**
 - **Les types composites**

Le type des colonnes

■ Les types énumération

- **Déclaration de types énumérés**

```
CREATE TYPE type_couleur AS ENUM ('bleu', 'vert', 'rouge');
```

- **Utilisation d'un type ENUM**

```
CREATE TABLE voiture  
( id serial primary key,  
  modele varchar(45),  
  couleur type_couleur  
);
```

```
INSERT INTO voiture VALUES (default, 'focus', 'rouge');  
INSERT INTO voiture VALUES (default, 'focus', 'bleu');  
INSERT INTO voiture VALUES (default, 'ibiza', 'vert');  
INSERT INTO voiture VALUES (default, 'focus', 'noir'); -- erreur
```

id	modele	couleur
1	focus	rouge
2	focus	bleu
3	ibiza	vert

Erreur SQL :

ERREUR: valeur en entrée invalide pour le enum type_couleur : « noir »

Le type des colonnes

■ Les types énumération

```
..... AS ENUM ('bleu', 'vert', 'rouge');
```

- Tri sur des types énumérés

```
SELECT * FROM voiture WHERE couleur > 'bleu';
```

id	modele	couleur
1	focus	rouge
3	ibiza	vert

```
SELECT * FROM voiture WHERE couleur > 'bleu' order by couleur;
```

id	modele	couleur
3	ibiza	vert
1	focus	rouge

```
SELECT * FROM voiture  
WHERE couleur = (SELECT MAX(couleur)FROM voiture);
```

id	modele	couleur
1	focus	rouge

Le type des colonnes

■ Les types tableaux

- Déclaration de type tableau

```
CREATE TABLE sal_emp
(
    nom text,
    paye_par_semaine integer[], -- tableau à une dimension
    planning text[][]           -- tableau à deux dimensions
);
```

- Insertion de valeurs dans un type tableau

```
INSERT INTO sal_emp VALUES ('Bill',
                              '{10000, 10000, 10000, 10000}',
                              '{{"rendez-vous", "repas"}, {"entraînement", "présentation"}}');

INSERT INTO sal_emp VALUES ('Carol',
                              '{20000, 25000, 25000, 25000}',
                              '{{"petit-déjeuner", "consultation"}, {"rendez-vous", "repas"}}');
```


Le type des colonnes

■ Les types tableaux

- Insertion de valeurs dans un type tableau

Avec le constructeur ARRAY

```
INSERT INTO sal_emp VALUES ('David',  
                             ARRAY[10450, 8300, 6700, 12340],  
                             ARRAY[['cinéma', 'revue projet'], ['production', 'diner']]);
```

- Affichage du contenu de la table

```
SELECT * FROM sal_emp;
```

nom	paye_par_semaine	planning
Bill	{10000,10000,10000,10000}	{{rendez-vous,repas},{entrainement,présentation}}
Carol	{20000,25000,25000,25000}	{{petit-déjeuner,consultation},{rendez-vous,repas}}
David	{10450,8300,6700,12340}	{{cinéma,"revue projet"},{production,diner}}

Le type des colonnes

■ Les types tableaux

- Accéder à un élément du contenu de la table
Attention les indices commencent à 1

```
SELECT paye_par_semaine, paye_par_semaine[1]
FROM sal_emp;
```

paye_par_semaine	paye_par_semaine
{10000,10000,10000,10000}	10000
{20000,25000,25000,25000}	20000
{10450,8300,6700,12340}	10450

```
SELECT planning,planning[1][2],planning[2][1]
FROM sal_emp
```

planning	planning	planning
{{rendez-vous,repas},{entrainement,présentation}}	repas	entrainement
{{petit-déjeuner,consultation},{rendez-vous,repas}}	consultation	rendez-vous
{{cinéma,"revue projet",{production,diner}}	revue projet	production

Pour en savoir plus sur la manipulation des types tableaux :

<http://docs.postgresqlfr.org/9.1/arrays.html>

Le type des colonnes

■ Création d'un domaine de valeur

Un domaine est essentiellement un type de données avec des contraintes optionnelles (restrictions sur l'ensemble de valeurs autorisées)

Syntaxe :

```
CREATE DOMAIN nom [AS] type_donnee  
    [ DEFAULT expression ]  
    [ constraint [ ... ] ]
```

où **constraint** est :

```
[ CONSTRAINT nom_contrainte ]  
{ NOT NULL | NULL | CHECK (expression) }
```

Le type des colonnes

■ Création d'un domaine de valeur

Exemple :

Définition

```
CREATE DOMAIN DOM_tel AS varchar(20)
    constraint tel_val
        check (VALUE SIMILAR TO '[0-9][0-9]-[0-9][0-9]-[0-9][0-9]-[0-9][0-9]-[0-9][0-9]');
```

Utilisation

```
CREATE TABLE client
(id_cli int primary key,
 nom_cli varchar(25),
 prenom_cli varchar(25),
 tel_cli DOM_tel );
```

Le type des colonnes

■ Les types composites

Un type composite s'apparente à une structure de données utilisés dans différents langages de programmation.

Création d'un type composé : syntaxe générale

```
CREATE TYPE nom_type AS (  
    nom_colonne1 { type },  
    nom_colonne2 { type }, ..... );
```

*Impossible de définir des contraintes sur les colonnes lors de la déclaration du type,
il faudra définir un domaine de valeur pour définir une contrainte*

Exemple

```
CREATE TYPE complexe AS  
( r real,  
  i real);
```

Le type des colonnes

■ Les types composites

Un type composite s'apparente à une structure de données utilisés dans différents langages de programmation.

Utilisation d'un type composé

- Exemple de déclaration de colonne avec le type composé:

```
CREATE TABLE nombre  
    ( nombre_C complexe,  
      nombre_Ent integer );
```

- Exemple d'insertion de données dans une colonne de type composé:

```
INSERT INTO nombre VALUES (ROW(0.56, 1.99), 3);
```

- Exemple de modification d'une colonne de type composé

```
UPDATE nombre  
SET nombre_c = ROW(0.75, 0.85) ;
```

Le type des colonnes

- Les types composites

Utilisation d'un type composé

Exemple d'affichage des colonnes du type composé :

```
SELECT (nombre_C).r as "Réal", (nombre_C).i as "Imaginaire"  
FROM nombre;
```

Réal	Imaginaire
0.75	0.85

A decorative vertical pattern on the left side of the slide, consisting of various colored hexagons (blue, yellow, black) and dots arranged in a somewhat abstract, upward-pointing shape.

Le type des colonnes

- Les types composites

Exemple de suppression du type composé :

Drop type complexe ;

Le type des colonnes

- Les types composites, définir une table de ce type

Exemple :

```
CREATE TYPE type_employe AS (id int, nom text, salaire numeric);
```

```
CREATE TABLE employes OF type_employe (  
    PRIMARY KEY (id),  
    nom WITH OPTIONS DEFAULT 'Dirlo',  
    salaire WITH OPTIONS DEFAULT 1000  
);
```

Il est impossible de définir des contraintes sur les colonnes d'un type composite lors de sa définition => WITH OPTIONS ...

Si on supprime le type composé on doit supprimer la table qui est de ce type : *DROP type_employe CASCADE*



Les contraintes

- Les contraintes de table

- La gestion de l'intégrité référentielle

- *Mode de gestion de la référence pour les clés composites*

Défini les règles de concordance entre les valeurs d'une clé étrangère composite(composée de plus d'une colonne) et les valeurs de la clé primaire référencée.

MATCH SIMPLE (correspondance simple)

autorise une colonne de clé étrangère à être NULL même si les autres parties de la clé étrangère ne sont pas nulles.

MATCH FULL (correspondance totale)

n'autorise une colonne d'une clé étrangère composite à être NULL que si l'ensemble des colonnes de la clé étrangère sont NULL

Les contraintes

■ Les contraintes de table

■ La gestion de l'intégrité référentielle

■ *Mode de gestion de la référence pour les clés composites*

```
CREATE TABLE FOURNITURE
(
  F      VARCHAR(3),
  P      VARCHAR(3),
  QTE    DECIMAL(5) CHECK (QTE > 0),
  FCT_MONTANT DECIMAL(16,2),
  CONSTRAINT PK_Fourn PRIMARY KEY (F, P) );
```

```
CREATE TABLE COMMANDE(
  C      varchar(3) primary key,
  F      varchar(3),
  P      varchar(3),
  QTE    decimal(5) check (qte > 0),
  CONSTRAINT FK_cde_four FOREIGN KEY (F, P) REFERENCES FOURNITURE(F,P)
  MATCH SIMPLE -- / FULL );
```

Dans la table fourniture, la clé primaire est composée de deux colonnes (F,P), la table commande fait référence à la table fourniture, comme la clé étrangère doit faire référence à la totalité de la clé primaire alors la clé étrangère est composée elle aussi de deux colonnes (F,P)

Les contraintes

■ Les contraintes de table

■ La gestion de l'intégrité référentielle

■ *Mode de gestion de la référence pour les clés composites*

```
INSERT INTO FOURNITURE VALUES ('f1', 'p1', 1, 234);
```

f	p	qte	fct_montant
f1	p1	1	234.00

INSERT INTO COMMANDE VALUES	SIMPLE	FULL
('c1', 'f1', 'p1', 1);	OK	OK
('c2', 'f1', 'p2', 1);	ERREUR	ERREUR
('c3', 'f6', NULL, 1);	OK	ERREUR
('c4', 'f1', NULL, 1);	OK	ERREUR
('c5', NULL, 'p1', 1);	OK	ERREUR
('c6', NULL, NULL, 1);	OK	OK

On s'intéresse à la 2 et 3ème valeur qui sont portées par les colonnes qui composent la clé étrangère

A vertical decorative element on the left side of the slide, composed of a cluster of small blue, yellow, and black hexagons.

MODIFICATION DE LA STRUCTURE D'UNE TABLE

A decorative element on the right side of the slide, consisting of a cluster of blue, yellow, and black hexagons, some of which are larger and more prominent than others.

Modification et Suppression

■ Ajout d'une colonne

Syntaxe :

```
ALTER TABLE relation  
    ADD COLUMN nom_colonne type [contrainte]
```

ou

```
ALTER TABLE relation  
    ADD nom_colonne type [contrainte]
```

Ajoutons l'attribut *fax* et l'attribut *ville*:

```
ALTER TABLE T_PERSONNE  
    ADD prs_fax DECIMAL(10,0),  
    ADD prs_ville varchar(10) ;
```

Les nouvelles colonnes sont placées à la fin de la table

Modification et Suppression

■ Supprimer une colonne

Attention, supprimer un attribut implique la suppression des valeurs qui se trouvent dans la colonne qui correspond à cet attribut.

Syntaxe :

```
ALTER TABLE relation DROP COLUMN attribut;
```

ou

```
ALTER TABLE relation DROP attribut;
```

Exemple :

```
ALTER TABLE T_PERSONNE DROP COLUMN prs_Ville;
```

Modification et Suppression

■ Ajout/suppression d'une contrainte de clé étrangère

Ajout :

```
ALTER TABLE T_TELEPHONE  
    ADD CONSTRAINT FK_TEL_CLI FOREIGN KEY (CLI_ID)  
    REFERENCES T_CLIENT(CLI_ID);
```

Suppression :

```
ALTER TABLE T_TELEPHONE  
    DROP CONSTRAINT FK_TEL_CLI;
```


Modification et Suppression

■ Ajout/suppression d'une clé primaire

Ajout :

```
ALTER TABLE Personnes ADD PRIMARY KEY (nom,prenom)
```

Supression :

```
ALTER TABLE Personnes DROP constraint Personnes_pkey ;
```

ATTENTION : Vous devez chercher dans le schéma de la base de données le nom de la contrainte qui défini la clé primaire.

Par défaut ce nom est composé du nom de la table suivi de 'pkey'

Exemple : si la clé est dans la tables *t_personne* alors la contrainte de clé primaire ce nomme *t_personne_pkey*

Modification et Suppression

■ Changer le nom et/ou le type d'une colonne

Changer de nom de la colonne CLI_PRENOM de la table T_CLIENT

```
ALTER TABLE T_CLIENT  
    RENAME COLUMN CLI_PRENOM TO PRENOM;
```

Changer le type de la colonne PRENOM sans la renommer

```
ALTER TABLE T_CLIENT  
    ALTER COLUMN PRENOM TYPE VARCHAR(20);
```

Modification et Suppression

■ Modifier/Supprimer la valeur par défaut d'une colonne

Modifie la valeur par défaut de la colonne CLI_ENSEIGNE de la table T_CLIENT

```
ALTER TABLE T_CLIENT  
    ALTER CLI_ENSEIGNE SET DEFAULT 'PARTICULIER';
```

Supprime la valeur par défaut de la colonne CLI_ENSEIGNE

```
ALTER TABLE T_CLIENT  
    ALTER CLI_ENSEIGNE DROP DEFAULT ;
```

Modification et Suppression

■ Renommer une table

```
ALTER TABLE T_CLIENT  
    RENAME TO CLIENT ;
```

Pour en savoir plus sur la modification de la structure d'une table :

<http://docs.postgresqlfr.org/8.3/sql-altertable.html>

■ Supprimer une table

```
DROP {TABLE | VIEW } NOM_OBJET ;
```

Exemple :

```
DROP TABLE personne ;
```



FIN