

Bases de données SQL

Tables et clés

BD102

v400c

2022-01-25



Christina.Khnaisser@USherbrooke.ca

Luc.Lavoie@USherbrooke.ca

© 2018-2021, Μητρετ (http://info.usherbrooke.ca/lavoie)
CC BY-NC-SA 4.0 (https://creativecommons.org/licenses/by-nc-sa/4.0/)

Plan

- Préambule
- CREATE
- DROP
- ALTER
- Un exercice avec Gaspard et Madeleine
- Perspectives
- Les colles du prof

Préambule

**Pour la notation
grammaticale, voir
SQL_02c**

- Portée

Portée de la présentation

Exemple CREATE TABLE PostgreSQL

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ] TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
  | table_constraint
  | LIKE source_table [ like_option ... ] }
```

and **table_constraint** is:

```
[ CONSTRAINT constraint_name ]
{ CHECK ( expression ) [ NO INHERIT ] |
  UNIQUE ( column_name [, ... ] ) index_parameters |
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |
  EXCLUDE [ USING index_method ] ( exclude_element WITH operator [, ... ] ) index_parameters [ WHERE ( predicate ) ] |
  FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn [, ... ] ) ]
  [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ] [ ON DELETE action ] [ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

<https://www.postgresql.org/docs/9.5/static/sql-createtable.html>

Ouf!

Dans un premier temps, nous présenterons donc

- *une grammaire simplifiée,*
- *sous-ensemble propre de la grammaire du dialecte PostgreSQL,*
- *le plus souvent conforme à la norme ISO 9075:2016,*
- *mais ne comprenant que les éléments essentiels.*

CREATE

- Créer (déclarer) un objet.
- Par exemple, une table
(une représentation approximative de relvar).

2022-01-25 MCIID_SQL_02b — Tables et clés (14/004) © 2018-2022, Mirtis — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

5

et implicitement de son type!

Ce type est accessible lors de la création d'une autre table en utilisant la syntaxe

`CREATE TABLE t1 (like t2)`

Le langage SQL
La commande CREATE

creation ::=

CREATE objet

objet ::=

objTable |

objVue |

objDomaine |

objType |

objAssertion |

<autres objets>

Création d'un schéma : définition des objets

Le langage SQL**CREATE TABLE - les colonnes**

objTable ::=

TABLE nomTable ({ defTable ... , })

defTable ::=

defColonne | defContrainte

defColonne ::=

nomColonne

type

[[**NOT**] **NULL**][**UNIQUE**][**DEFAULT** valeur][**CHECK** (condition)] -- voir SQL_04-Operateurs-et-expressions

Le langage SQL

nomColonne

- Un nomColonne désigne un identifiant d'attribut de la table (donc un identifiant d'attribut des tuples qu'elle contient).
- Un nomColonne est dénoté par un identificateur.
- Il existe deux formes d'identificateurs :
 - la forme simple,
 - la forme délimitée.

Le langage SQL

Identificateurs – forme simple

- **Forme simple (sans délimiteur)**
 - **lettre (lettre | chiffre | autre) ***
 - on ne peut utiliser que les lettres, les chiffres et quelques autres symboles *variant d'un dialecte à un autre* (mais comprenant au moins le tiret bas « _ »);
 - relativement à la casse des lettres, en conformité avec la norme ISO, le SDBD doit produire une forme interne normalisée (*malheureusement non prescrite*);
 - la forme normalisée varie donc d'un SGBD à l'autre :
 - en majuscules (Oracle, DB2, etc.);
 - en minuscules (PostgreSQL, MySQL, SQLite, etc.).

Les identificateurs dont la forme est simple se comportent donc, entre eux, comme s'ils étaient insensibles à la casse.

La non-prescription de la représentation normalisée entraîne des confusions en présence d'identificateurs sous la forme délimitée.

Le langage SQL

Identificateurs – forme délimitée

- **Forme délimitée (donc avec délimiteurs)**
 - tout symbole peut être utilisé;
 - la forme normalisée coïncide avec la suite de symboles elle-même;
 - le délimiteur varie selon le dialecte
 - guillemets (ISO, ANSI, Oracle, PostgreSQL, DB2, MariaDB...)
 - "Élève"
 - crochets (Microsoft, T-SQL)
 - [Élève]

Les identificateurs sous forme délimitée sont donc différenciables selon la casse, les accents, les esprits, les formes liminaires, etc.

Le langage SQL

Contextualisation des identificateurs

- En regard des tables d'un même schéma
 - Table.colonne
- En regard des tables de schémas différents
 - Schema.table.colonne

Le langage SQL

Contraintes d'attribut

- La seule contrainte d'attribut que vous **devez** utiliser est :
NOT NULL
- Il est toujours préférable d'exprimer les autres contraintes d'attributs comme des contraintes de table, ce qui permet de les nommer.
- En général, la pseudo-contrainte DEFAULT induit plus de mal que de bien, car des erreurs peuvent ainsi passer inaperçues, surtout au gré de l'évolution des tables.

Concernant le « style » recommandé dans le cadre des travaux du cours, voir le module BD190.

Le langage SQL**CREATE TABLE — contraintes**

defContrainte ::=

```
[CONSTRAINT nomContrainte]
{
  CHECK (condition) -- voir SQL_04-Operateurs-et-expressions
| PRIMARY KEY ( listeNomsColonne )
| UNIQUE ( listeNomsColonne )
| cléRéférentielle
}
```

cléRéférentielle ::=

```
FOREIGN KEY ( listeNomsColonne )
REFERENCES nomTable [ ( listeNomsColonne ) ]
[ MATCH { SIMPLE | PARTIAL | FULL } ]
[ ON UPDATE action ]
[ ON DELETE action ]
```

action ::=

```
CASCADE | SET NULL | SET DEFAULT | NO ACTION
```

2022-01-25

MCIÉD_SQL_02b — Tables et clés (v4f0d4) © 2018-2022, MFCIS — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

13

Extraits de la norme ISO 9075:2011

The checking of a constraint depends on its constraint mode within the current SQL-transaction. If the constraint mode is immediate, then the constraint is effectively checked at the end of each SQL-statement.

NOTE 29 — This includes SQL-statements that are executed as a direct result or an indirect result of executing a different SQL-statement.

If the constraint mode is deferred, then the constraint is effectively checked when the constraint mode is changed to immediate either explicitly by execution of a <set constraints mode statement>, or implicitly at the end of the current SQL-transaction.

A referential constraint is satisfied if one of the following conditions is true, depending on the <match type> specified in the <referential constraint definition>:

- If no <match type> was specified then, for each row R1 of the referencing table, either at least one of the values of the referencing columns in R1 shall be a null value, or the value of each referencing column in R1 shall be equal to the value of the corresponding referenced column in some row of the referenced table.
- If MATCH FULL was specified then, for each row R1 of the referencing table, either the value of every referencing column in R1 shall be a null value, or the value of every referencing column in R1 shall not be null and there shall be some row R2 of the referenced table such that the value of each referencing column in R1 is equal to the value of the corresponding referenced column in R2.
- If MATCH PARTIAL was specified then, for each row R1 of the referencing table, there shall be some row R2 of the referenced table such that the value of each referencing column in R1 is either null or is equal to the value of the corresponding referenced column in R2.

NOTE 30 — If MATCH FULL or MATCH PARTIAL is specified for a referential constraint and if the referencing table has only one column specified in <referential constraint definition> for that referential constraint, or if the referencing table has more than one specified column for that <referential constraint definition>, but none of those columns is nullable, then the effect is the same as if no <match type> were specified.

Le langage SQL

Note sur les clés (1)

- La première clé doit être déclarée
 - PRIMARY KEY
- Les autres clés doivent être déclarées
 - UNIQUE

- On peut ne déclarer aucune clé, mais, *contrairement à ce que prévoit la théorie relationnelle*, l'ensemble des attributs ne formera pas une clé pour autant (*puisque que SQL utilise une sémantique de collection plutôt que d'ensemble*).
- En pratique, il est donc très fortement recommandé de déclarer au moins une clé.

Pourquoi cette différence entre primary et unique ? Pour des raisons historiques liées à l'inférence (automatisée) du schéma physique dans le contexte technologique des années 1970. Ce contexte n'est plus d'actualité depuis les années 1980, mais la distinction est demeurée dans le langage pour des raisons de rétrocompatibilité. Les informaticiens sont TRÈS conservateurs.

L'absence de déclaration d'une clé peut parfois être justifiée dans un contexte d'importation ou de migration de données (dans un ELT ou un ETL, par exemple).

Le langage SQL

Note sur les clés (2)

- Attribut **PRIMARY KEY**
 - ne peut être nul
- Attribut **UNIQUE**
 - peut être nul !!!
- L'annulabilité totale ou partielle d'une clé entraîne de nombreux problèmes.
- Il convient donc de ne **jamais** permettre l'annulabilité d'un attribut participant à une clé.
- La source du problème réside en l'impossibilité de comparer deux attributs dont au moins est nul. À suivre...

DROP

- Retirer un objet.
- Par exemple, une table.

Le langage SQL**La commande DROP**

Retrait ::=

DROP objet

objet ::=

objTable | objType | <autres objets>

objTable ::=

TABLE nomTable comportement

comportement ::=

[**RESTRICT** | **CASCADE**]**Particularité du dialecte Oracle :**

comportement ::=

[**CASCADE CONSTRAINTS**]

DROP TABLE Étudiant CASCADE

ALTER

- Modifier la structure (représentation) d'un objet.
- Par exemple, une table.

Le langage SQL

La commande ALTER (1/2)

Modification ::=

ALTER objet

objet ::=

objTable | objType | <autres objets>

objTable ::=

TABLE nomTable { modColonne | modContrainte }

Le langage SQL**La commande ALTER (2/2)**

modColonne ::=

ADD [COLUMN] defColonne

| **ALTER [COLUMN] nomColonne modAction**

| **DROP [COLUMN] nomColonne**

modAction ::=

modDefault | modNULL | <autres modifications>

modDefault ::=

SET DEFAULT valeur | DROP DEFAULT

modNULL ::=

SET NOT NULL | DROP NOT NULL

modContrainte ::=

ADD defContrainte

| **DROP CONSTRAINT nomContrainte**

defColonne et defContrainte
comme dans CREATE TABLE

<alter column action> ::=

<set column default clause> | <drop column default clause>

| <set column not null clause> | <drop column not null clause>

| <add column scope clause> | <drop column scope clause>

| <alter column data type clause> | <alter identity column specification>

| <drop identity property clause> | <drop column generation expression clause>

<http://www.postgresql.org/docs/9.4/static/sql-altertable.html>

Exemples :

-- Renommer la colonne adresse par ville

ALTER TABLE Etudiant RENAME adresse TO ville;

-- Ajouter une colonne à la table résultat pour attribuer une mention : I (incomplet), R (réussi), E (échec)

ALTER TABLE Resultat ADD mention CHAR(1);

-- Modifier le type de la colonne note en numeric

ALTER TABLE Resultat ALTER note TYPE NUMERIC(3,2) – attention à la problématique de conversion des valeurs!

Le langage SQL Et la suite ?

- Nous pouvons créer, modifier et détruire des tables.
- Nous pouvons déclarer les clés candidates d'une table.
- Nous pouvons déclarer les clés référentielles entre deux tables.

- Qu'en est-il des autres contraintes à l'intérieur d'une table?
 - Voir le module BD103-LDD-03
- Qu'en est-il des contraintes générales sur plusieurs tables ?
 - Voir le module BD110-LDD-05

Exemples

- Évaluation
- Gaspard et Madeleine

Évaluation – rappels Schéma relationnel

- Nous avons déjà utilisé le schéma Évaluation lors de la présentation du module BD012 pour illustrer la théorie relationnelle.
- Voir le module BD100 pour sa « traduction » en SQL.

**Gaspard et Madeleine
schéma relationnel simplifié**

DefTypeArme

(typeArme)

Arme

(noProduit, nomProduit, typeArme, poids)

DefTypeMunition

(typeMunition)

Munition

(noProduit, nomProduit, typeMunition)

Armement

(typeArme, typeMunition)

Stock

(noProduit, quantite)

...

UnicitéProduit

 $(Arme \pi \{noProduit\}) \cap (Munition \pi \{noProduit\}) = \emptyset$

Faire la représentation graphique au tableau

Gaspard et Madeleine Au travail !

```
CREATE TABLE Arme (  
  noProduit CHAR(6),  
  nomProduit VARCHAR(80),  
  typeArme VARCHAR(15),  
  poids NUMERIC(4)  
);
```

Mais ce n'est pas si simple, implicitement SQL considère tous les attributs comme étant annulables...

Gaspard et Madeleine

Itération 0 – annulation de l'annulabilité 😊

```
CREATE TABLE Arme (  
  noProduit CHAR(6) NOT NULL,  
  nomProduit VARCHAR(80) NOT NULL,  
  typeArme VARCHAR(15) NOT NULL,  
  poids NUMERIC(4) NOT NULL  
);
```

Où sont les clés ?

Gaspard et Madeleine

Itération 1 – déclaration des clés

```
CREATE TABLE Arme (  
  noProduit CHAR(6) NOT NULL,  
  nomProduit VARCHAR(80) NOT NULL,  
  typeArme VARCHAR(15) NOT NULL,  
  poids NUMERIC(4) NOT NULL,  
  CONSTRAINT Arme_cc0 PRIMARY KEY (noProduit),  
  CONSTRAINT Arme_cc1 UNIQUE (nomProduit)  
);
```

Gaspard et Madeleine Discussion

- Pouvons-nous mettre n'importe quel type d'armes ou ne faut-il pas s'assurer qu'il fait partie d'un ensemble déterminé ?
 - Voir version 2

Gaspard et Madeleine

Itération 2

```
CREATE TABLE DefTypeArme (  
  typeArme    VARCHAR(15) NOT NULL,  
  CONSTRAINT  DefTypeArme_cc0 PRIMARY KEY (typeArme)  
);  
CREATE TABLE Arme (  
  noProduit   CHAR(6)      NOT NULL,  
  nomProduit  VARCHAR(80)  NOT NULL,  
  typeArme    VARCHAR(15) NOT NULL,  
  poids       NUMERIC(4)  NOT NULL,  
  CONSTRAINT  Arme_cc0 PRIMARY KEY (noProduit),  
  CONSTRAINT  Arme_cc1 UNIQUE (nomProduit),  
  CONSTRAINT  Arme_cc1  
    FOREIGN KEY (typeArme) REFERENCES DefTypeArme  
);
```

Faire remarquer

- * l'identité de définition de typeArme dans les deux tables;
- * le problème que causerait un typeArme nul, et ce, dans les deux sens.

Gaspard et Madeleine
Discussion (bis)

- Nous avons trouvé une solution technique, mais n'aurions-nous pas dû revenir à la modélisation au préalable ?
 - Voir Gaspard et Madeleine !

Références

- Le site de PostgreSQL (en français)
 - <http://docs.postgresqlfr.org>
- Gaspard et Madeleine
 - http://info.usherbrooke.ca/llavoie/enseignement/Modules/BD011-Gaspard-et-Madeleine_NDC.pdf

