# Bases de données
# SQL

## Tables et clés

BD102
v400c

2022-01-25

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
Luc.Lavoie@USherbrooke.ca

# 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).

## Le langage SQL
## La commande CREATE

creation ::=

CREATE objet

objet ::=

objTable |

objVue |

objDomaine |

objType |

objAssertion |

<autres 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.).

## 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]

## 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.

## 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
```

## 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.

## 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 entraine 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 ]

# 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 ::=
    modDefaut | modNULL | <autres modifications>

modDefaut ::=
    SET DEFAULT valeur | DROP DEFAULT

modNULL ::=
    SET NOT NULL | DROP NOT NULL

modContrainte ::=
    ADD defContrainte
  | DROP CONSTRAINT nomContrainte

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

## 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_ce1
      FOREIGN KEY (typeArme) REFERENCES DefTypeArme
);
```

## 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