Bases de données
SQL

Tables et clés

SQL_02c
v401c
2025-01-20

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

Plan

- Préambule
- CREATE
- DROP
- ALTER
- Perspectives
- Les colles du prof

Préambule

- Portée

Pour la notation grammaticale, voir SQL_02a

Portée de la présentation
Syntaxe du CREATE TABLE en 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:

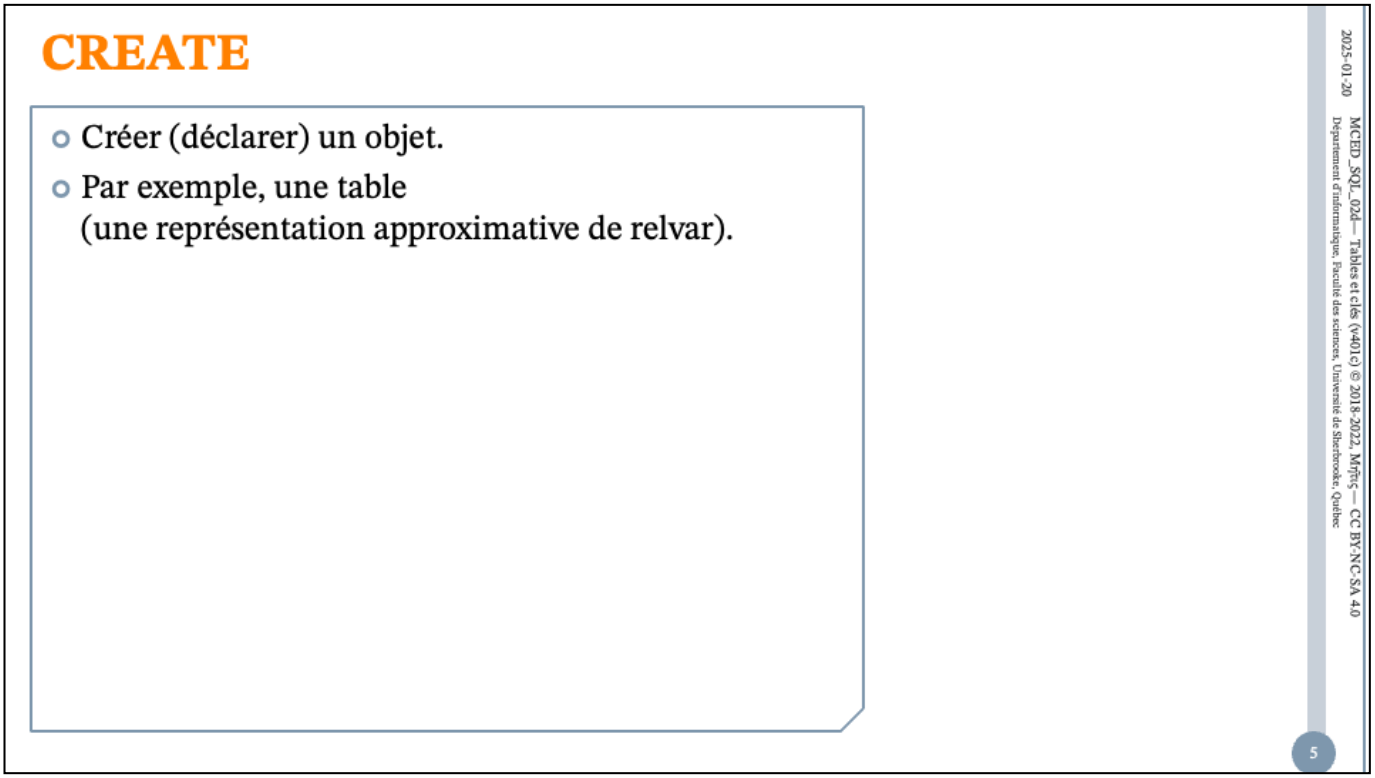
```
[ CONSTRAINT constraint_name ]
{ CHECK ( expression ) [ NO INHERIT ] |
  UNIQUE ( column_name [, ... ] ) index_parameters |
  PRIMARY KEY ( column_name [, ... ] ) index_parameters |
  EXCLUDE [ USING index_method ] ( exclude_element WITH operator [, ... ] ) index_parameters [ WHERE ( predicate ) ] |
  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 de la norme ISO 9075:2016,*
- *avec, au besoin, certaines particularités de PostgreSQL.*



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 et les contraintes

objTable ::=
    TABLE *nomTable* **(** { *defTable* ... **,** } **)**
defTable ::=
    defColonne | *defContrainte*
defColonne ::=
    nomColonne
    type
    [[**NOT**] **NULL**]
    [**UNIQUE**]
    [**DEFAULT** *valeur*]
    [**CHECK (***condition***)**]
        -- pour *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 entraine 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 différentiables selon la casse, les accents, les esprits, les formes liminaires, etc.

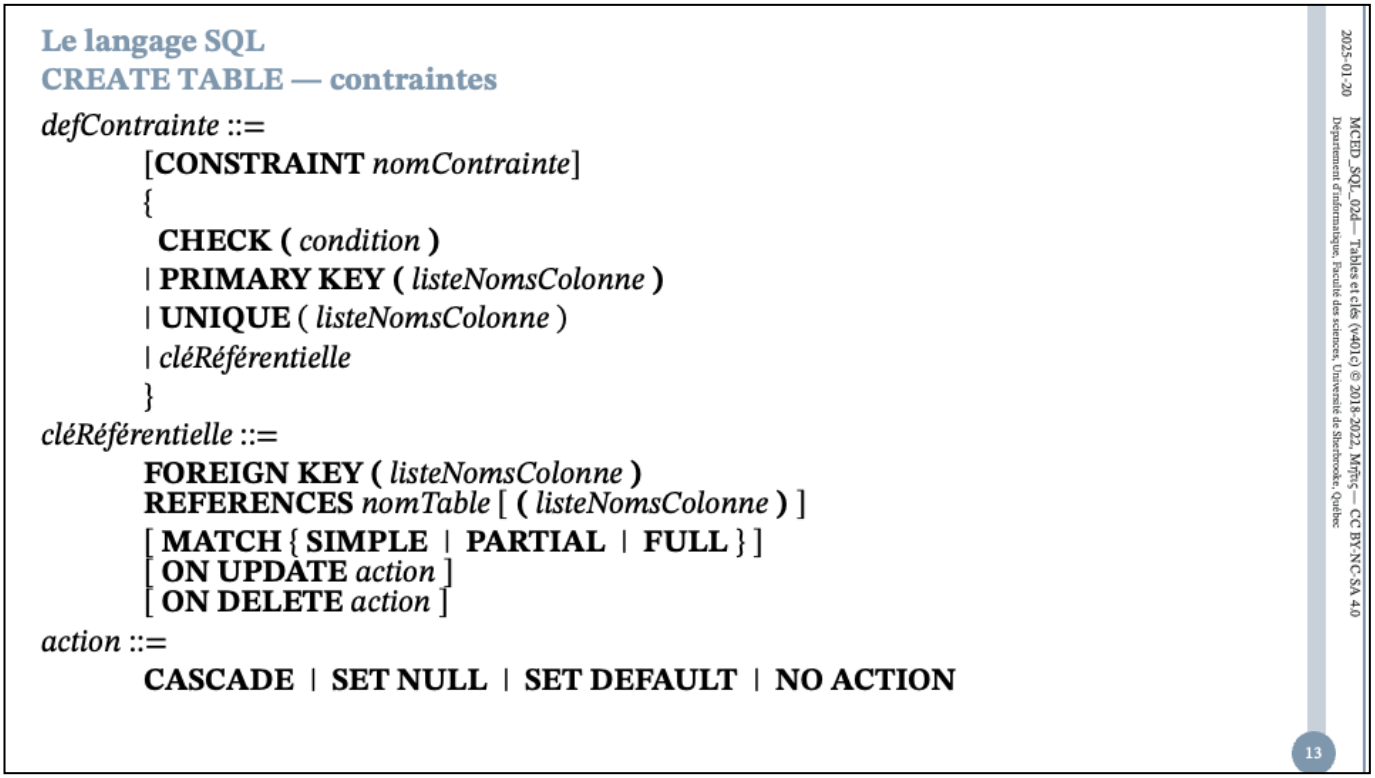
Le langage SQL
Contextualisation des identificateurs

- 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.



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 candidate doit être déclarée
  - **PRIMARY KEY**
- Les autres clés candidates doivent être déclarées
  - **UNIQUE**

- On peut ne déclarer aucune clé candidate, 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 *toujours* 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)

- Un attribut **PRIMARY KEY**
  - ne peut être nul
- Un 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 | *objDomain* | *objType* | *autres_objets*

objDomain ::=
TABLE *nomTable comportement*

objTable ::=
DOMAIN *nomDomaine comportement*

objTable ::=
TYPE *nomType 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*

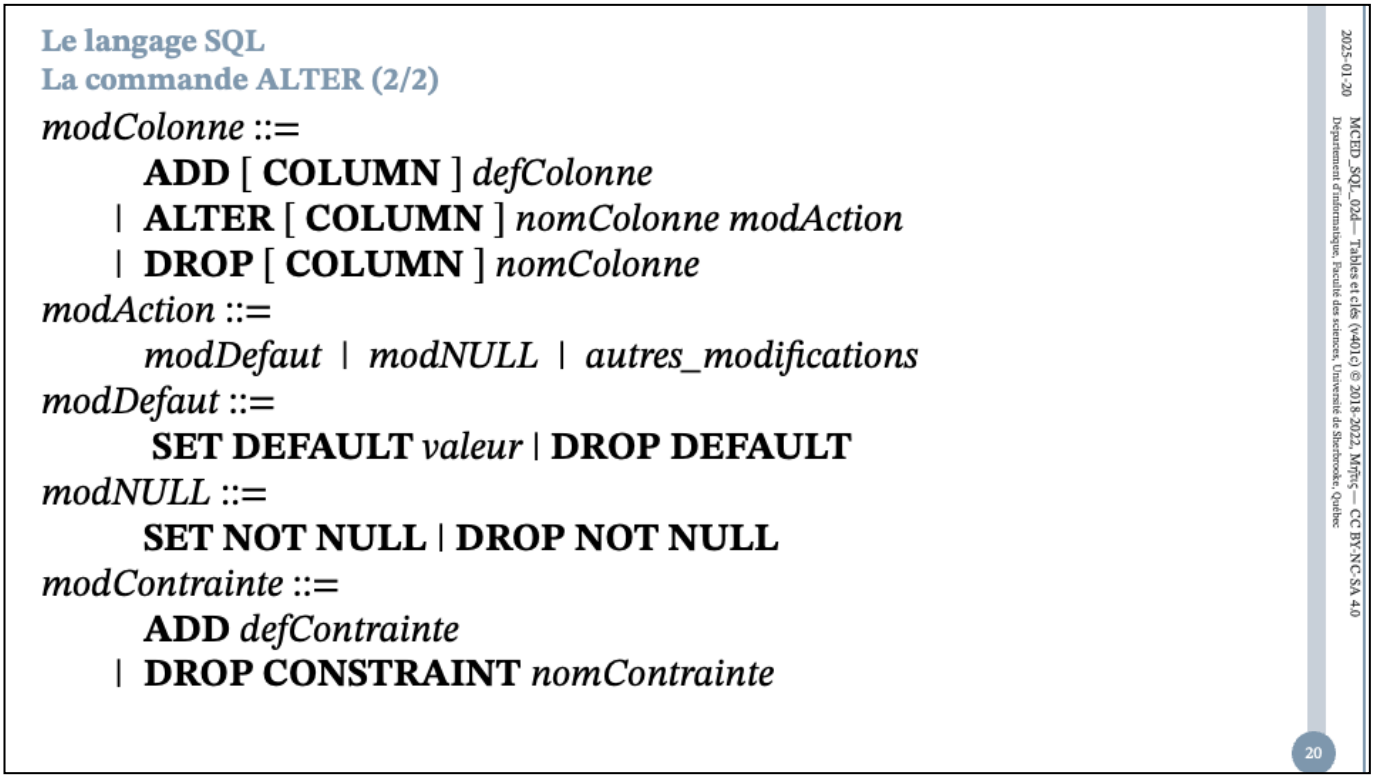
objet ::=

objTable | *objType* | *autres_objets*

objTable ::=

TABLE *nomTable* { *modColonne* | *modContrainte* }

Voir la documentation de PostgreSQL pour les spécificités de **ALTER** dans les cas *objDomain* et *objType*



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 SQL_03
- Qu'en est-il des contraintes générales sur plusieurs tables ?
  - Voir le module SQL_05

Exemples

- Évaluation
- Films
- Gaspard et Madeleine

Exemples disponibles sur le site de cours

Suggestion
Utiliser un atelier de développement (*) pour faciliter l'analyse du code

(*) pgAdmin4 , Eclipse, DataGrip, Navicat, etc.

Références

- Le site de PostgreSQL (en français)
  - http://docs.postgresqlfr.org
  - plus particulièrement le chapitre 15, rubrique CREATE TABLE
    https://doc.postgresql.fr/17/sql-createtable.html