Bases de données
SQL

Modification
(Insert, Delete, Update)

SQL_04
v303b
2025-01-30

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

Plan

- INSERT
- DELETE
- UPDATE
- Exercices
- Incidences sur les clés référentielles
- Références
- Les colles du prof

Le Langage SQL
INSERT (syntaxe)

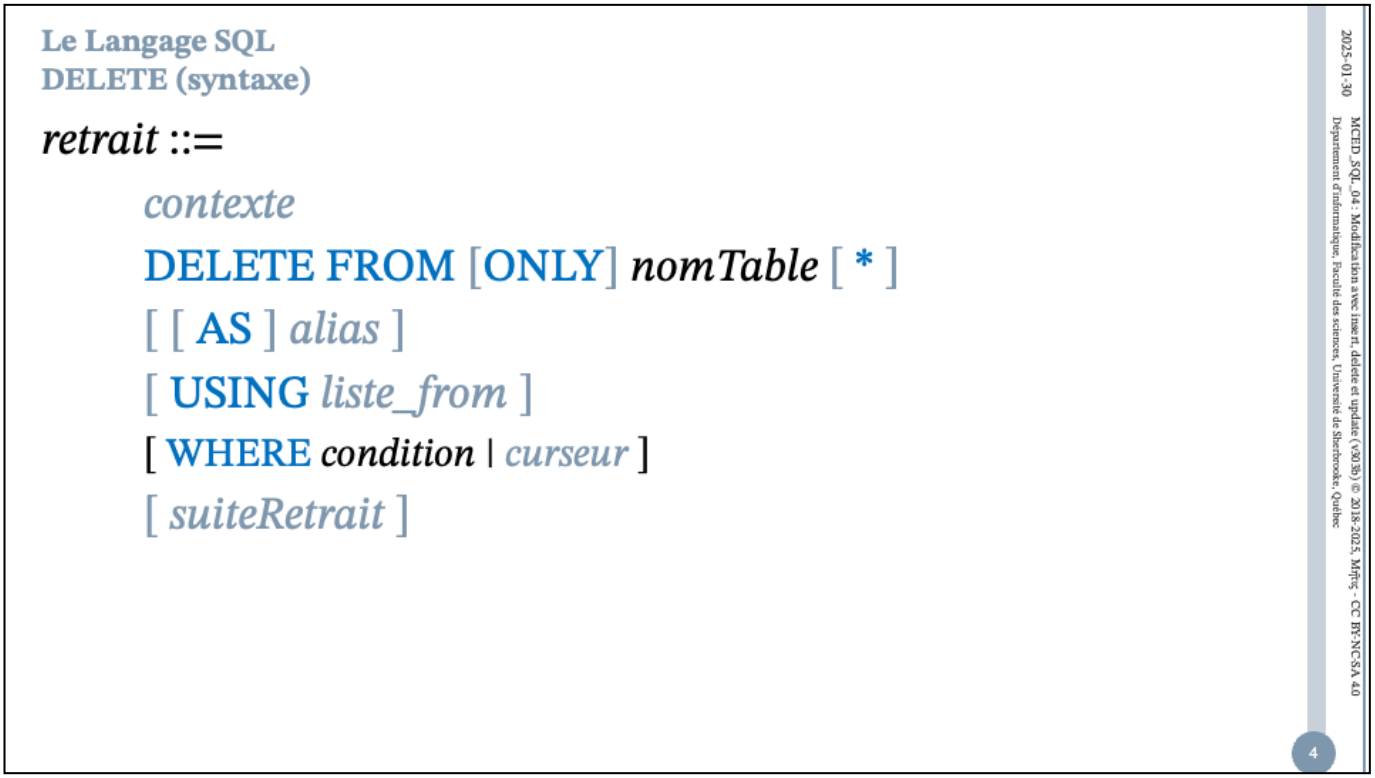
insertion ::=
    contexte
    INSERT INTO *nom_table*
    [(*listeNomsColonne*)]
    {
      DEFAULT VALUES
    | VALUES *listeExpressionInit*
    | *requête*
    }
    [*suiteInsertion*]

listeExpressionInit ::=
    { *tuple-val* ... , }

tuple-val ::=
    ({ { *expression* | DEFAULT } ... , })

Ajout de tuple :
- Ajout d'un seul tuple
- Ajout d'un tuple à partir d'une requête
- Ajouts multiples

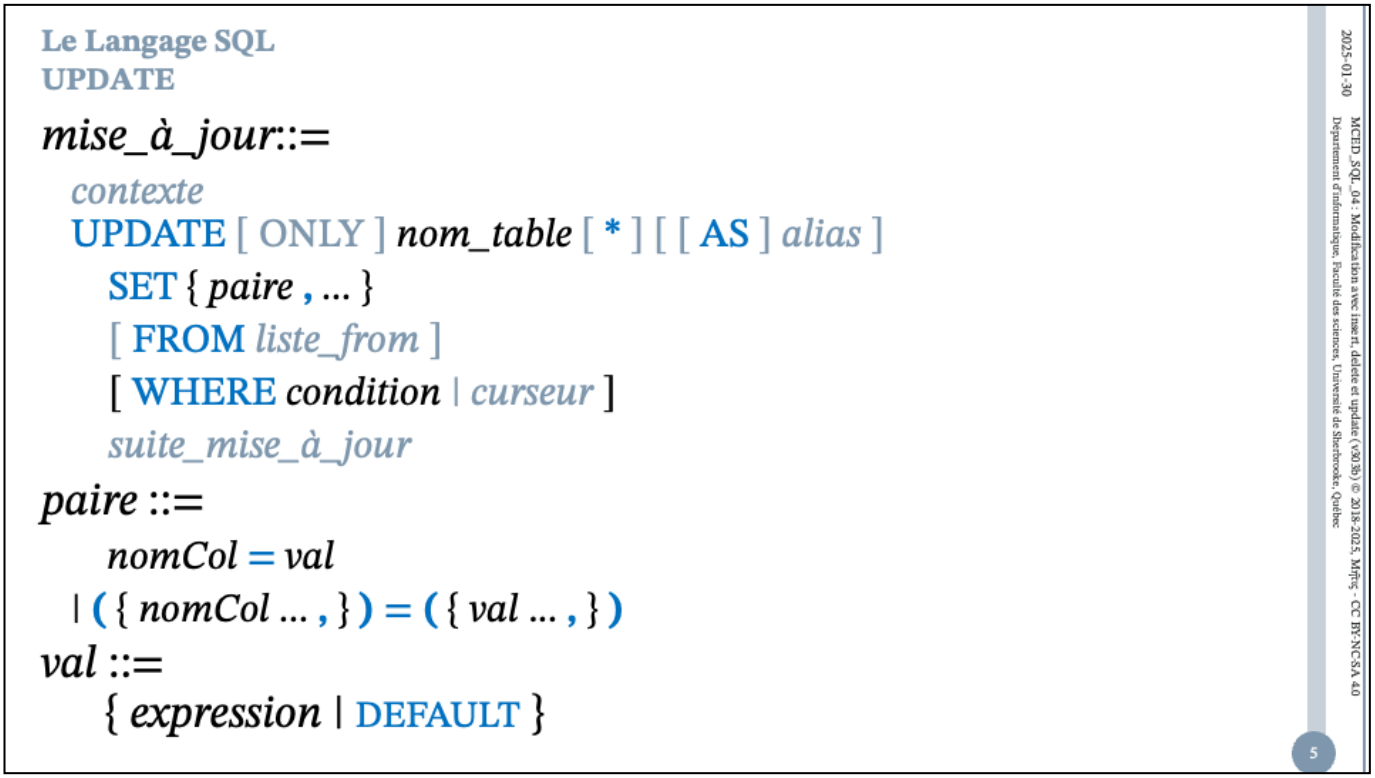
Il est fortement recommandé de toujours donner explicitement la liste des colonnes.



La clause USING a un rôle analogue au FROM du select... sauf que
 * elle est limitée au seuls produits cartésiens (pas de jointure) ;
 * la table nomTable en fait déjà partie et sera jointe à la liste_from.
Corolairement, si on désire en fait une jointure, il faut l'exprimer par une condition au sein de la clause WHERE.

contexte ::=
   [WITH [RECURSIVE] requête_with [, ...]]
curseur ::=
   WHERE CURRENT OF nom_curseur
suiteRetrait ::=
   [RETURNING * | expression_sortie [[AS] nom_sortie] [, ...]]
liste_from ::=
   item_from [, ...]

Pour plus de détails
   https://www.postgresql.org/docs/current/sql-delete.html



La clause FROM de UPDATE a un rôle analogue au FROM du SELECT... sauf que
* elle est limitée au seuls produits cartésiens (pas de jointure) ;
* la table nomTable en fait déjà partie et sera jointe à la liste_from.
Corolairement, si on désire en fait une jointure, il faut l'exprimer par une condition au sein de la clause WHERE.

contexte ::=
[WITH [RECURSIVE] requête_with [, ...]]
curseur ::=
WHERE CURRENT OF nom_curseur
suite_mise_à_jour ::=
[RETURNING * | expression_sortie [[AS] nom_sortie] [, ...]]
liste_from ::=
item_from [, ...]

Pour plus de détails, voir
https://www.postgresql.org/docs/15/sql-update.html

Exemples

- Évaluation
- Gaspard et Madeleine (à venir)

Évaluation – rappels

Exemple de données

Activité

sigle	titre
IFT 159	Analyse et programmation
IFT 187	Éléments de bases de données
IMN 117	Acquisition des médias numériques
IGE 401	Gestion de projets
GMQ 103	Géopositionnement

TypeÉvaluation

code	description
IN	Examen intra
FI	Examen final
TP	Travail pratique
PR	Projet

Résultat

matricule	TE	activité	trimestre	note
15113150	TP	IFT 187	20133	80
15112354	FI	IFT 187	20123	78
15113150	TP	IFT 159	20133	75
15112354	FI	GMQ 103	20123	85
15110132	IN	IMN 117	20123	90
15110132	IN	IFT 187	20133	45
15112354	FI	IFT 159	20123	52

Étudiant

matricule	nom	adresse
15113150	Paul	ᐳᕕᖏᓂᖅ
15112354	Éliane	Blanc-Sablon
15113870	Mohamed	Tadoussac
15110132	Sergeï	Chandler

Évaluation
Insertion

- Ajouter un nouveau type d'évaluation

```
INSERT INTO TypeEvaluation (code, description)
  VALUES ('TD', 'Travail dirigé');
```

- Ajouter une nouvelle activité au programme

```
INSERT INTO Activite (sigle, titre)
  VALUES ('IFT339', 'Structure de données');
```

- Ajouter le résultat de l'examen final de Paul

```
INSERT INTO Resultat
  VALUES ('15113150', 'FI', 'IFT187', '20133', 75);
-- L'omission de la liste des colonnes n'est pas
-- une bonne idée, pourquoi ?
```

Le Langage SQL
INSERT (exemples)

- INSERT INTO Resultat VALUES – êtes-vous certain de l'ordre ?
  ('15113150', '20133', 'IFT187', 'TP', 80);

- INSERT INTO Resultat
  (matricule, activite, trimestre, TE, note)
  VALUES
  ('15113150', 'IFT187', '20133', 'TP', 80);

- INSERT INTO Resultat
  (matricule, activité, trimestre, TE, note)
  VALUES
  ('15112354', 'IFT187', '20123', 'FI', 78),
  ('15113150', 'IFT159', '20133', 'TP', 75),
  ('15112354', 'GMQ103', '20123', 'FI', 85),
  ('15110132', 'IMN117', '20123', 'IN', 90),
  ('15110132', 'IFT187', '20133', 'IN', 45),
  ('15112354', 'IFT159', '20123', 'FI', 52);

-- Ajouter un nouveau type d'évaluation

```
INSERT INTO TypeEvaluation(code, description)
  VALUES ('TD', 'Travail dirigé');
```

-- Ajouter une nouvelle activité au programme

```
INSERT INTO Activite(sigle, titre)
  VALUES ('IFT339', 'Structure de données');
```

-- Ajouter le résultat de l'examen final de Paul

```
INSERT INTO Resultat
  VALUES ('15113150', 'FI', 'IFT187', '20133', 75)
```

Évaluation
Retrait

- Retirer les activités IMN du catalogue

```
DELETE FROM Activite
  WHERE SUBTR(sigle,1,3) = 'IMN';
```

- Retirer le type d'évaluation TP

```
DELETE FROM TypeEvaluation
  WHERE code = 'TP';
```

- Retirer les notes des TP

```
DELETE FROM Resultat
  WHERE TE = 'TP';
```

Le Langage SQL
DELETE (exemples)

- `DELETE FROM Resultat ;`
- `DELETE FROM Resultat WHERE trimestre = '20123' ;`

Retirer l'activité IMN du catalogue

```
DELETE FROM Activite
  WHERE SUBTR(sigle, 1,3) = 'IMN';
```

Retirer le type d'évaluation TP

```
DELETE FROM TypeEvaluation
  WHERE code = 'TP';
```

Quel serait le résultat ?

Retirer les notes des TP

```
DELETE FROM Resultat
  WHERE TE = 'TP';
```

Évaluation
Mise à jour

- Ajouter 10% pour tous les étudiants

```
UPDATE Resultat SET note = note + (note * 0,1);
```

ou encore

```
UPDATE Resultat SET note = note * 1,1;
```

- Retirer 5 points à la note de TP de tous les étudiants de l'activité 'IFT187'

```
UPDATE Resultat SET note = note – 5
WHERE TE = 'TP' AND activite = 'IFT187';
```

Le Langage SQL
UPDATE (exemples)

```
UPDATE Resultats
  SET note = 90
WHERE matricule ='15113150'
  AND activite = 'IFT187'
  AND trimestre = '20133'
  AND TE = 'TP'
;
```


-- Ajouter 10% pour tous les étudiants

```
UPDATE Resultat SET note = note * 0,1;
```

-- Retirer 5 points à tous les étudiants de l'activité 'IFT187'

```
UPDATE Resultat SET note = note – 5
FROM Etudiant
WHERE activite = 'IFT187';
```

-- Retirer 5 points à Éliane pour son cours 'IFT187'

```
UPDATE Resultat SET note = note – 5
FROM Etudiant
WHERE nom = 'Éliane'
  AND etudiant.matricule = resultat.matricule
 AND activite = 'IFT187';
```

Évaluation
Mise à jour avec jointure

Retirer 5 points à l'examen final d'Éliane pour le cours 'IFT187'

```
UPDATE Resultat
  SET note = note - 5
FROM Etudiant
WHERE Etudiant.matricule=Resultat.matricule
  AND nom = 'Éliane'
  AND activite = 'IFT187'
  AND TE = 'FI';
```

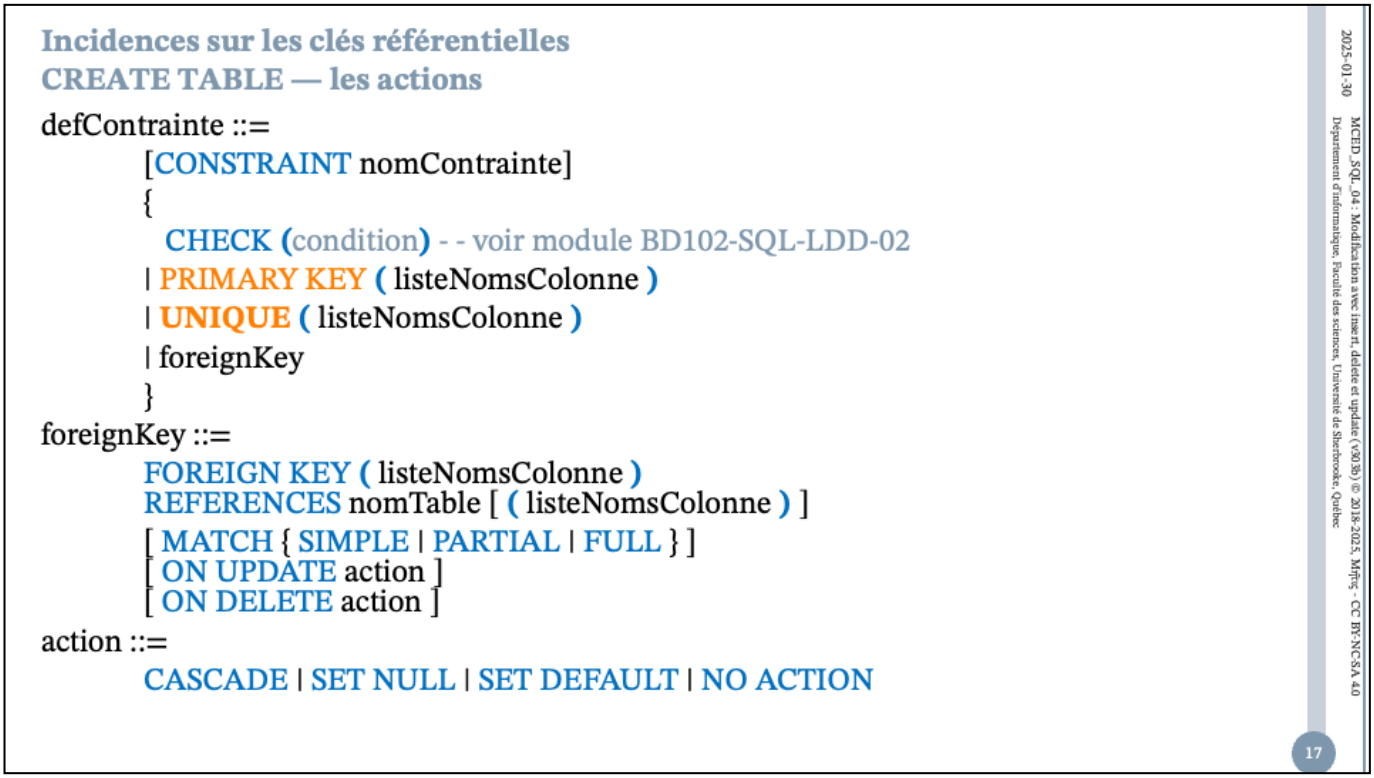

Que se passe-t-il si Éliane à suivi le cours deux fois en faisant chaque fois l'examen final ?

Incidence sur les clés référentielles

- Que peut-il se passer ?

Que peut-il se passer ?

- Que se passe-t-il quand
  - un tuple référé est détruit ?
  - un tuple référé est modifié ?
  - un tuple référent est détruit ?
  - un tuple référent est modifié ?



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.

Incidences sur les clés étrangères
Note (1/2)

- Les termes

  [ON UPDATE action]
  [ON DELETE action]

  permettent de spécifier une action lorsqu'une clé de la contrainte référentielle est modifiée. Les actions possibles sont

  CASCADE | SET NULL | SET DEFAULT | NO ACTION

- L'action ainsi spécifiée est appliquée aux tuples dépendants consécutivement à une modification du tuple référé.

Le fait que les attributs référés forment une clé de la table référée, simplifie grandement la description du comportement et sa mise en oeuvre; aussi SQL en a-t-il fait une obligation.

Incidences sur les clés référentielles
Note (2/2)

Le terme

[MATCH { SIMPLE | PARTIAL | FULL }]

permet de contrôler le traitement des NULL lors de la comparaison des clés :

- si vous avez suivi nos conseils, tous vos attributs (a fortiori ceux participant à une clé) sont NOT NULL... c'est donc inutile;
- sinon **devez** prendre en compte cet extrait de la norme ISO :

- 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.

Références

- Elmasri et Navathe (4e ed.), chapitre 7
- Elmasri et Navathe (6e ed.), chapitre 4
- [Date2012]
  Date, Chris J. ;
  SQL and Relational Theory: How to Write Accurate SQL Code.
  2nd edition, O'Reilly, 2012.
  ISBN 978-1-449-31640-2.
- Le site d'Oracle (en anglais)
  - http://docs.oracle.com/cd/E11882_01/index.htm
- Le site de PostgreSQL (en français)
  - http://docs.postgresqlfr.org