Bases de données
SQL

Expressions
Opérateurs scalaires

SQL_03a
v401e
2025-01-26

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

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

Plan

- Expressions et conditions
- Opérateurs prédéfinis élémentaires
  - Booléens
  - Nombres
  - Textes
  - Autres opérateurs
- Exercices
- Les colles du prof

© 2018-2025, MrTs — CC BY-NC-SA 4.0

Expressions et conditions

- Expression
- Condition

Définitions

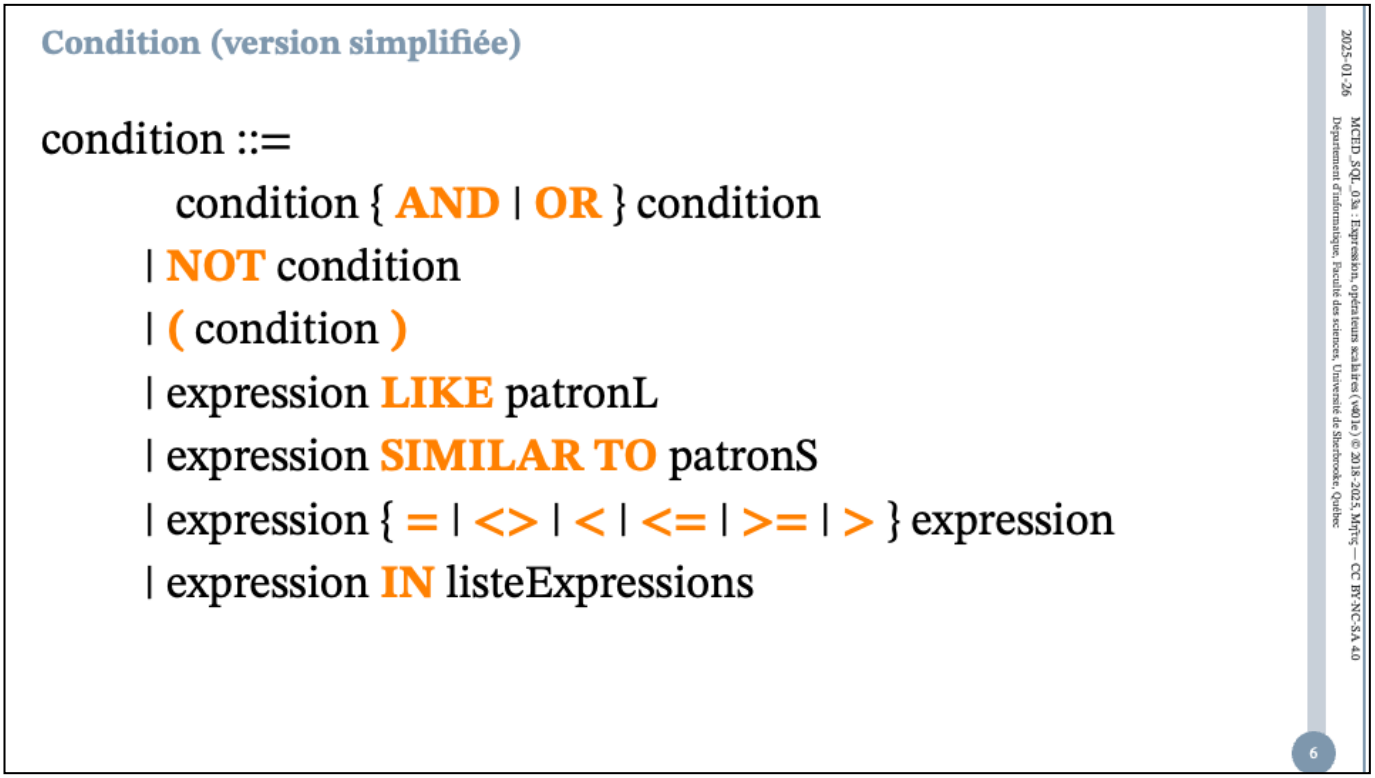
- Qu'est donc qu'une expression ?
  - La dénotation d'une suite d'opérations (portant sur des variables) dont l'évaluation résulte en une valeur typée.

- Qu'est donc qu'une condition ?
  - Tout simplement une expression dont l'évaluation résulte en une valeur booléenne.

Expression (version simplifiée)

```
expression ::=
        expression  opBinaire expression
      | opUnaire expression
      | nomFonction listeExpressions
      | ( expression )
      | condition
      | attribut
      | constante
      | VALUE
opBinaire ::=
        + | - | * | / | % | ||          -- et quelques autres
opUnaire ::=
        + | -                           -- et quelques autres
listeExpressions ::=
        ( { expression ... , } )
```

2025-01-26 MCED_SQL_03a : Expression, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec



Les tests à l'aide de l'opérateur IS, par exemple

| attribut **IS** [**NOT**] **NULL**

sont reportés au module traitant des informations manquantes.

```
CREATE TABLE Activite (
  sigle         CHAR(6) NOT NULL,
  titre         VARCHAR(46) NOT NULL,
  CONSTRAINT Activite_cc0 PRIMARY KEY (sigle),
  CONSTRAINT Activite_sigle0 CHECK
      ( sigle SIMILAR TO '[A-Z]{3}[0-9]{3}' )
  CONSTRAINT Activite_sigle1 CHECK
      ( SUBSTR(sigle,1,3) IN ('IFT', 'IMN', 'IGL', 'GMQ', 'MAT') )
  );
```

Remarques

- les deux contraintes peuvent être combinées en une seule ;
- la contrainte Activite_sigle1 est une fausse bonne idée, il est préférable de scinder le sigle en deux parties (la discipline et le numéro de cours) et de recenser les disciplines indépendamment dans une table propre. Pouvez-vous dire pourquoi ?

Fonction

- Qu'est donc qu'une fonction ?
  - La dénotation d'une suite d'opérations (portant sur des paramètres) dont l'évaluation résulte en une valeur typée.
  - La valeur de chaque paramètre est déterminée par une expression elle-même évaluée lors de l'appel de la fonction.
- Deux catégories de fonctions
  - définies par l'utilisateur
    - *à l'aide d'un sous-langage conçu à cet effet*
  - prédéfinies
    - *par la norme SQL*
      - les principales (mais pas la plupart) sont présentées ci-après
    - *par le dialecte SQL reconnu par un SGBD spécifique*
      - se reporter à la documentation de l'éditeur du SGBD
      - faire attention à la non-transportabilité

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Opérateurs et fonctions

- Une différence de syntaxe, une identité de rôle.
- Opérateurs et fonctions : même combat !

© 2018-2025, MrTs — CC BY-NC-SA 4.0

Fonctions et opérateurs prédéfinis

- Fonctions et opérateurs logiques
- Fonctions et opérateurs numériques
- Fonctions et opérateurs textuels
- Fonctions et opérateurs temporels
- Fonctions opérateurs des langages rationnels
  - de reconnaissance (similar)
  - d'extraction (substr)
  - et une forme historique simplifiée (like)
- ...

Fonctions et opérateurs logiques

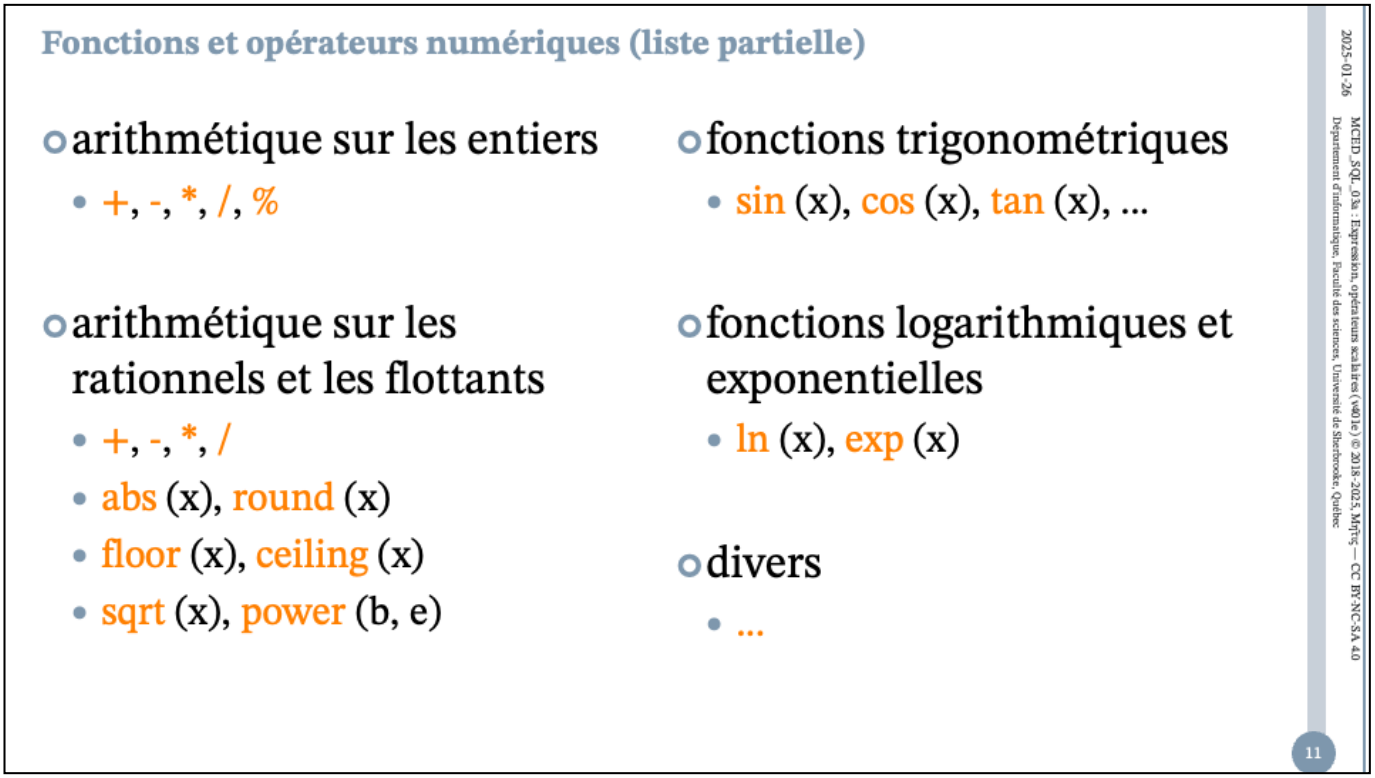
OR	true	false	unknown
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

AND	true	false	unknown
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

P	NOT P
true	false
false	true
unknown	unknown

IS	true	false	unknown
true	true	false	false
false	false	true	false
unknown	false	false	true

2025-01-26 MCED_SQL_03a : Expression, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec



Il existe une bonne centaine de fonctions et d'opérateurs supplémentaires.

Pour la différence entre mod et rem, voir https://fr.wikipedia.org/wiki/Modulo_(opération)
Vérifier ce que dit la norme SQL.

Sauriez-vous comment obtenir le log10 ?

Fonctions et opérateurs textuels (liste partielle)

- char_length (*s*)
- octet_length (*s*)
- *s1* || *s2*
- substr (*s* [from *d*] [for *l*])
- trim ([[leading | trailing | both] [*c*] from] *s*)
- position (*s*, *t*)
- overlay (*s1* placing *s2* from *d* [for *l*])
- upper (*s*)
- lower (*s*)
- convert (*s* using *x*)
- translate (*s* using *x*)
- *... et une bonne centaine d'autres !*

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

https://www.postgresql.org/docs/11/functions-string.html

Fonctions et opérateurs temporels prédéfinis (liste partielle)

- current_timestamp
- current_date
- current_time
- make_timestamp
- make_date
- make_time
- make_interval
- extract(*field* from *x*)
  - *field* := **YEAR** | **MONTH** | **DAY** | **HOUR** | **MINUTE** | **SECOND**
  - pour *x* une valeur de type *timestamp, date, time* ou *interval*

- age(ts0, ts1) -- différence entre ts0 et ts1
- age(ts) -- équivalent à age(current_timestamp, ts)

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Prendre aussi en considération :
* transaction_timestamp
* statement_timestamp
* clock_timestamp()

Faut-il aussi introduire extract (epoch from ts) ?

Fonctions opérateurs des langages rationnels

- Introduction
- Syntaxe
- Conformité (similar)
- Extraction (substr)
- Forme historique simplifiée (like)
- Différences entre SQL et PostgreSQL
- Différences entre SQL et POSIX
- Exemples
- Exercices

Fonctions opérateurs des langages rationnels

- Les langages rationnels (LR) permettent de décrire et d'analyser les suites de symboles à l'aide d'un noyau d'opérateurs.
- Les LR sont équivalents aux automates d'états finis.
- Leur étude est approfondie par les activités
  - MAT 115 – mathématiques discrètes,
  - IFT 313 – introduction aux langages formels.
- Nous nous contenterons ici d'en décrire l'utilisation au sein du langage SQL.

Note
Les langages rationnels sont aussi parfois appelés langages réguliers, voire expressions régulières (par calque de l'anglais).

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Syntaxe des LR en SQL (1/6)

expR ::=

	atome	
|	*expR fermeture*	-- la fermeture de l'expression expR
|	$expR_1$ $expR_2$	-- $expR_1$ suivi de $expR_2$
|	$expR_1$ *alter* $expR_2$	-- $expR_1$ ou $expR_2$
|	(*expR*)	-- l'expression expR elle-même

alter ::=

|

Syntaxe des LR en SQL (2/6)

atome ::=

	_	-- représente n'importe quel caractère
|	%	-- représente n'importe quelle suite de caractères
|	^	-- représente le début d'une phrase
|	$	-- représente la fin d'une phrase
|	*caractère*	
|	*ensemble*	
|	*classe*	

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

'_' : représente n'importe quel caractère (alors que c'est le point dans le standard POSIX : « . »)
'%' : représente n'importe quelle suite de caractères (alors que c'est naturellement « .* » dans le standard POSIX).

Syntaxe des LR en SQL (3/6)

fermeture ::=

?	-- 0 ou 1 fois
| +	-- 1 ou plusieurs fois
| *	-- 0, 1 ou plusieurs fois
| { *entier* }	-- exactement *entier* fois
| { *entier* , }	-- au moins *entier* fois
| { $entier_1$, $entier_2$ }	-- entre $entier_1$ et $entier_2$ fois

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Syntaxe des LR en SQL (4/6)

caractère ::=

    caractère-simple | *caractère-codé*

caractère-simple ::=

    « tout caractère sauf ceux-ci _%^$[+*{\|() »

caractère-codé ::=

    \ *code*

hex ::=

    « une suite de chiffres hexadécimaux »

entier ::=

    « une suite de chiffres décimaux »

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Syntaxe des LR en SQL (5/6)

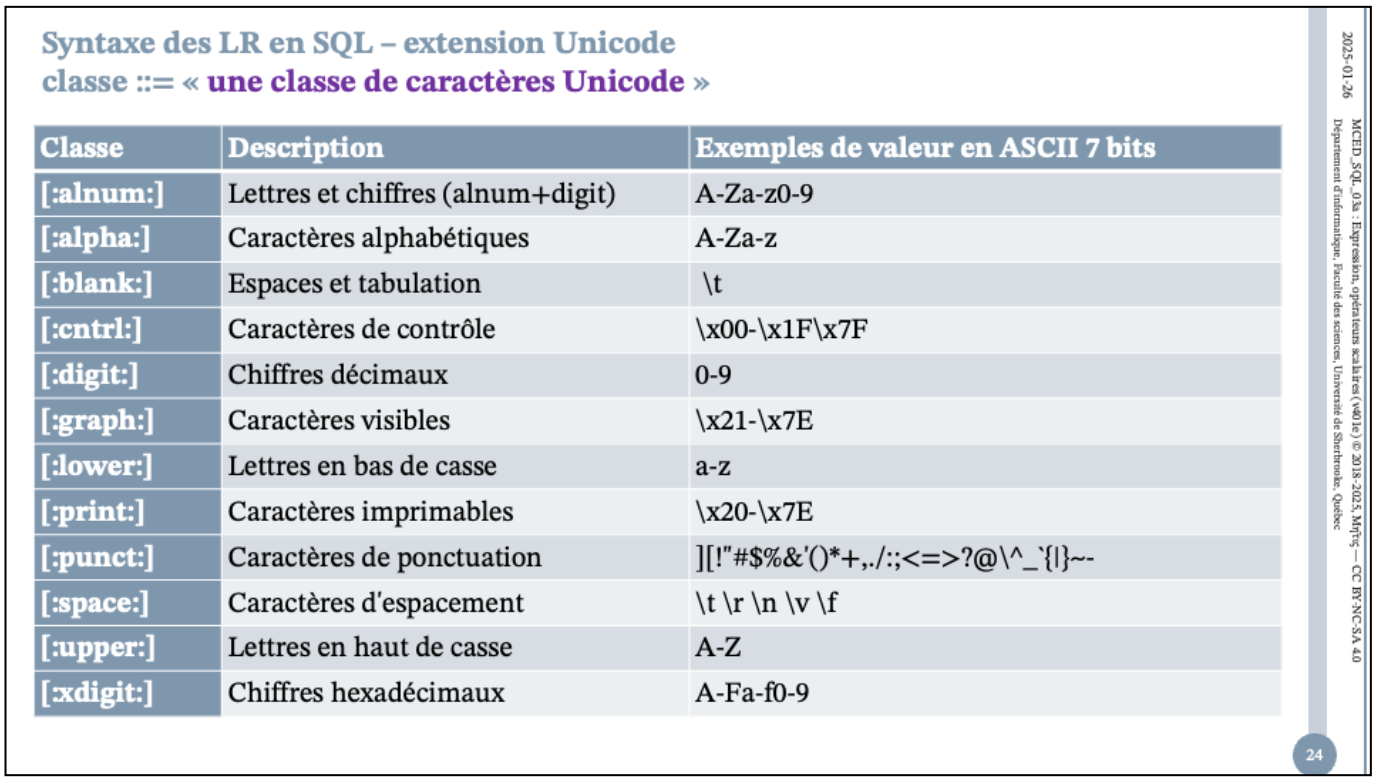
code ::=

	t	-- la commande HT (tabulation horizontale)
|	**r**	-- la commande CR (retour à la ligne)
|	**n**	-- la commande LF (ligne suivante)
|	**v**	-- la commande VT (tabulation verticale)
|	**f**	-- la commande FF (page suivante)
|	**x** *hex*	-- le car. Unicode de point de code hex
|	*entier*	-- le car. Unicode de point de code entier
|	« tout autre caractère sauf le : »	-- ce caractère lui-même

Syntaxe des LR en SQL (6/6)

ensemble ::=
    ensemble-simple | *ensemble-complémentaire*
ensemble-simple ::=
    [*suite-sans-crochet*]
ensemble-complémentaire ::=
    [^ *suite-sans-crochet*]
suite-sans-crochet ::=
    « toute suite de caractères sans crochet fermant, mais pouvant comprendre des intervalles représentés par deux caractères réunis par un tiret »

© 2018-2025, MrTs — CC BY-NC-SA 4.0



Syntaxe des LR en SQL – extension Unicode

classe ::= « **une classe de caractères Unicode** »

Classe	Description	Exemples de valeur en ASCII 7 bits
[:alnum:]	Lettres et chiffres (alnum+digit)	A-Za-z0-9
[:alpha:]	Caractères alphabétiques	A-Za-z
[:blank:]	Espaces et tabulation	\t
[:cntrl:]	Caractères de contrôle	\x00-\x1F\x7F
[:digit:]	Chiffres décimaux	0-9
[:graph:]	Caractères visibles	\x21-\x7E
[:lower:]	Lettres en bas de casse	a-z
[:print:]	Caractères imprimables	\x20-\x7E
[:punct:]	Caractères de ponctuation	][!"#$%&'()*+,./:;<=>?@\^_`{|}~-
[:space:]	Caractères d'espacement	\t \r \n \v \f
[:upper:]	Lettres en haut de casse	A-Z
[:xdigit:]	Chiffres hexadécimaux	A-Fa-f0-9

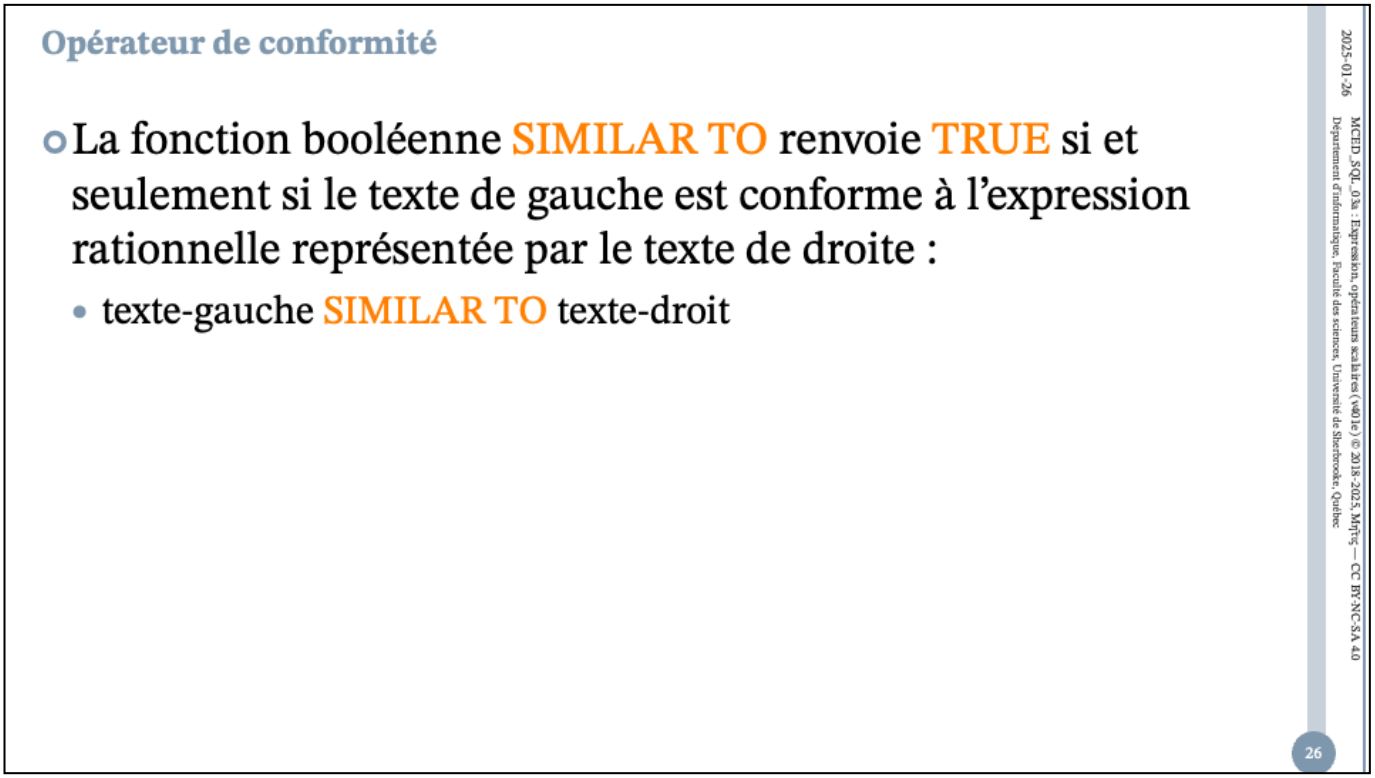
À l'intérieur d'une expression entre crochets, le nom d'une classe de caractères entouré entre [: et :] signifie la liste de tous les caractères appartenant à cette classe. Une classe de caractères ne peut pas être utilisée comme point final d'un intervalle. Le standard POSIX définit les noms de ces classes de caractères : alnum (lettres et chiffres), alpha (lettres), blank (espace et tabulation), cntrl (caractères de contrôle), digit (chiffres), graph (caractères affichages, sauf l'espace), lower (lettres minuscules), print (caractères affichages, incluant l'espace), punct (ponctuation), space (tout espace blanc), upper (lettres majuscules), et xdigit (chiffres hexadécimaux). Le comportement de ces classes de caractères standards est généralement cohérent sur les plateformes pour les caractères de l'ensemble ASCII 7 bits. Qu'un caractère non ASCII donné appartienne ou non à une de ces classes dépend de la *collation* utilisée par la fonction ou l'opérateur de l'expression rationnelle, (voir **Section 24.2**), ou par défaut de la locale indiquée par LC_CTYPE pour cette base (voir **Section 24.1**). La classification de caractères non ASCII peut varier entre les plateformes même pour des locales de nom similaire. (Mais la locale C ne considère jamais tout caractère non ASCII comme appartenant à une de ces classes.) En plus de ces classes de caractères standard, PostgreSQL définit the word character class, which is the same as alnum plus the underscore (_) character, and la classe de caractères ascii, qui contient l'ensemble ASCII 7 bits exactement.

Syntaxe des LR en SQL – une synthèse des opérateurs

- | représente une alternative (un choix) ;
- * représente la répétition des éléments précédents, 0 ou plusieurs fois ;
- + représente la répétition des éléments précédents, une ou plusieurs fois ;
- ? dénote une répétition du précédent élément zéro ou une fois ;
- {m} dénote une répétition du précédent élément exactement m fois ;
- {m,} dénote une répétition du précédent élément m ou plusieurs fois ;
- {m,n} dénote une répétition du précédent élément au moins m et au plus n fois ;
- les parenthèses (...) peuvent être utilisées pour grouper des éléments en un seul élément logique ;
- une expression entre crochets [...] spécifie un ensemble de caractères.

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

LIKE et SIMILAR : http://www.postgresql.org/docs/9.6./static/functions-matching.html



Les expressions rationnelles SQL sont un curieux mélange de la notation LIKE (introduite dans les versions initiales de la norme ANSI) et de la notation POSIX (communes aux systèmes de type Unix).

LIKE et SIMILAR : https://www.postgresql.org/docs/current/static/functions-matching.html

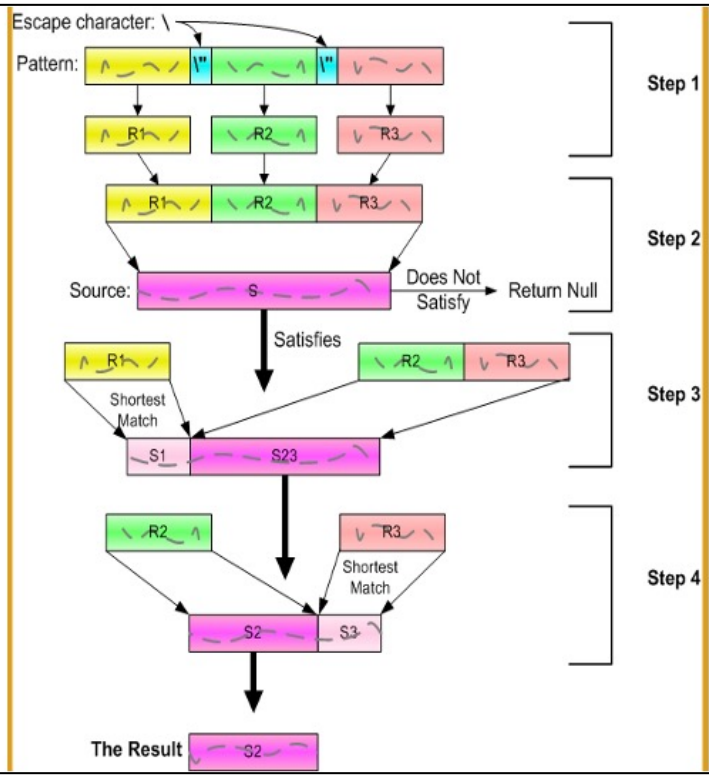
Opérateur d'extraction

- substr (*s* similar to *p* [escape *c*])

- Il est possible de segmenter une ER en trois parties : préfixe, infixe et suffixe de façon à isoler la seule partie infixe (voir diapositive suivante) à des fins d'extraction ou de remplacement.
- Les parties sont séparées par le caractère d'échappement c (*escape c*).

Algorithme utilisé par l'opérateur d'extraction

ISO/IEC 9075-2:2003 (E)
p. 18



Forme historique simplifiée

s [not] like *p* [escape *c*]

- La forme historique date de l'époque où les algorithmes d'évaluation des expressions rationnelles (ER) n'étaient pas abouties.
- Les normalisateurs de SQL ont donc brider volontairement les ER afin permettre une analyse simple en temps *O(n)*.
- Boyer et Moore ont publié en 2003 un algorithme optimal, valable pour toute ER.
- Il n'y a donc plus lieu d'avoir deux notations.

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Différences entre SQL et PostgreSQL

- Le dialecte PostgreSQL met à disposition de nombreuses abréviations, comme
  - ~ l'analogue à SIMILAR TO pour les ER dont la syntaxe est POSIX
- et de nombreuses fonctions supplémentaires
  - regexp_replace
  - regexp_matches
  - regexp_split_to_table
  - etc.

- Voir
  - https://docs.postgresql.fr/14/functions-matching.html

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Différences entre SQL et POSIX

- SQL utilise _ (tiret bas) et % (pourcentage) comme caractères de substitution représentant respectivement un caractère quelconque et une suite de caractères quelconques.
- POSIX utilise . (point) et .* (point suivi d'un astérisque) en lieu et place.
- Notez que le point n'est pas un méta-caractère pour SQL (SIMILAR TO).
- Notez que le _ et % ne sont pas des méta-caractères pour POSIX (~).
- On trouvera une bonne introduction à la notation POSIX dans
  https://fr.wikipedia.org/wiki/Expression_rationnelle

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Pour les détails :
https://www.postgresql.org/docs/current/functions-matching.html

Exemples

```
CONSTRAINT Activite_sigle CHECK (
  sigle SIMILAR TO '[A-Z]{3}[0-9]{3}'
  AND
  SUBSTR(sigle, 1,3) IN {'IFT', 'IMN', 'IGE', 'GMQ'}
)

CONSTRAINT Patient_NAM CHECK (
  NAM SIMILAR TO '[A-Z]{4}[0-9]{10}'
)

TYPE Identifiant
  Text
  CHECK (VALUE SIMILAR TO '[[:alpha:]][-_[:alnum:]]*')
```

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

-- Restreindre le catalogue des activité à celles d'informatique et de géomatique

Exercices

- Récrire la contrainte **`Activite_sigle`** en utilisant qu'une expression régulière (ne pas utiliser SUBSTR).
- Écrire une contrainte pour valider des numéros de téléphone canadiens.
- Écrire une contrainte pour valider des numéros de téléphone internationaux.

Autres opérateurs

- GREATEST, LEAST
- CAST
- CASE

2025-01-26 MCED_SQL_03a : Expression, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

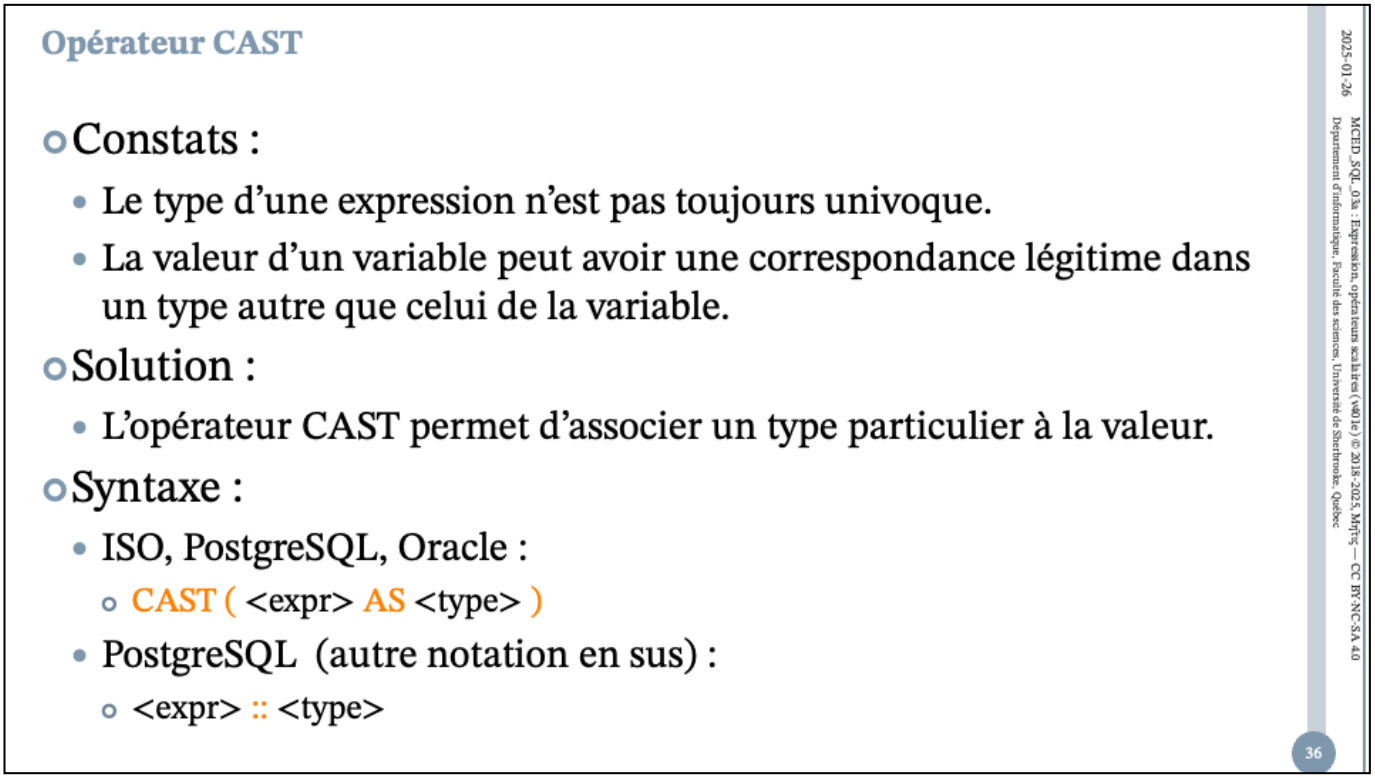
GREATEST, LEAST

- Pour une collection de valeurs d'un type ordonné
  - la fonction GREATEST retourne la valeur la plus grande,
  - la fonction LEAST retourne la valeur la plus petite.

- Exemple
  - 120 = GREATEST (100, 2, 40, 120, 8*5-1)
  - 2 = LEAST (100, 2, 40, 120, 8*5-1)
  - GREATEST (a*b, c+50, (e-f)+char_length(s))

Dans plusieurs langages de programmation, geatest et least sont plutôt désignés par max et min.
Or les fonctions max et min existent aussi en SQL, elles désignent plutôt des fonctions d'aggrégation – un sujet à venir.

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec



Un code de produit est représenté par 12 chiffres et la représentation choisie par le concepteur est NUMERIC(12)
Les produits congelés sont reconnaissables par le fait que les 4 derniers chiffres forment un nombre plus grand que 5000

```
CHECK
 (
  substr(CAST(code AS TEXT), 9, 4) > '5000'
 )
```

ou

```
CHECK
 (
  CAST (substr(CAST(code AS TEXT), 9, 4) AS INTEGER) > 5000
 )
```

Existe-t-il un autre moyen n'utilisant que des opérations arithmétiques ?

Opérateur CASE

- C'est un *opérateur* de choix (*pas une instruction*).
- C'est-à-dire qu'il permet de choisir une valeur sur la base d'une catégorisation établie à l'aide d'une condition (*simple case*) ou d'une énumération de valeurs (*searched case*).
- Il en existe donc deux formes

  <case expression> ::=

  <simple case> | <searched case>
- Dans tous les cas, l'opérateur retourne une valeur.

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Opérateur CASE – « simple »

<simple case> ::=
CASE {<simple when clause>...}
[**ELSE** <result>]
END
<simple when clause> ::=
WHEN <condition> **THEN** <result>
<result> ::=
<expression> | NULL

Contraintes :

- Tous les résultats doivent appartenir au même type.
- Si plus d'une condition est satisfaite, la première est choisie.
- En l'absence de clause ELSE, si aucune condition n'est satisfaite, la « valeur » NULL est retournée!

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

```
-- Valider la réussite au premier cycle
-- somme est la somme des cotes des cours terminés et n le nombre de cours réussis

CHECK
 (
  CASE
   WHEN n <> 0 THEN somme/n >= 2.0
   ELSE true
  END
 )
```

Opérateur CASE – « valué »

<searched case> ::=
CASE <expression> { <searched when clause> ... }
[**ELSE** <result>]
END
<searched when clause> ::=
WHEN <searched operand> **THEN** <result>
<searched operand> ::=
{ <expression> ... , }+

Contraintes :

- Tous les résultats doivent appartenir au même type.
- Si la valeur de l'expression est présente dans plus d'une liste, la première occurrence est choisie.
- En l'absence de clause ELSE, si la valeur de l'expression n'apparait dans aucune liste, la « valeur » NULL est retournée!

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

```
-- Valider la réussite selon le cycle
-- somme est la somme des cotes des cours terminés et n le nombre de cours réussis

CHECK
 (
  CASE
   WHEN n <> 0 THEN
    somme/n >=
          CASE cycle
             WHEN 1 THEN 2.0
             WHEN 2, 3 THEN 2.7
             ELSE 3.0
           END
   ELSE
    true
  END
 )
```

Exemples

- Évaluation
- Films
- Entrepôt

Exemples disponibles sur le site de cours

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

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

Évaluation – Activité
Contrainte sur sigle

```
CREATE TABLE Activite (
  sigle        TEXT NOT NULL,
  titre        TEXT NOT NULL,
  CONSTRAINT Activite_cc0 PRIMARY KEY (sigle),
  CONSTRAINT Activite_sigle0 CHECK
      (sigle SIMILAR TO '[A-Z]{3}[0-9]{3}')
  );

ALTER TABLE Activite
  ADD CONSTRAINT Activite_sigle1 CHECK
    (
      SUBSTR(sigle, 1,3)
        IN ('IFT', 'IMN', 'IGE', 'GMQ')
    );
```

Évaluation – Résultat
Contrainte sur note et trimestre

```
CREATE
  TABLE Resultat
  (
     matricule    TEXT NOT NULL,
     activite     TEXT NOT NULL,
     trimestre    TEXT NOT NULL,
     TE           TEXT NOT NULL,
     note         SMALLINT NOT NULL,
     […] ,
     CONSTRAINT Resultat_note CHECK
      ((0 <= note) AND (note <= 100)),
     CONSTRAINT Resultat_trimestre CHECK
      (
        (trimestre SIMILAR TO '[0-9]{4}[1-3]{1}')
      AND
        (SUBSTR (trimestre, 1, 4) >= '1956')
      )
  );
```

2025-01-26 MCED_SQL_03a : Expressions, opérateurs scalaires (v40.1e) © 2018-2025, MrTs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec


```
-- Les trimestres sont encodés en suffixant le chiffre du trimestre
-- à l'année. L'Université de Samarcande a été fondée en 1956.
```

Les colles du prof

- SQL comporte une fonction **mod** et un opérateur **%** tous deux réputés correspondre au reste de division entière.
- Par ailleurs, en mathématiques, comme dans plusieurs langages de programmation, on distingue le **reste** de la division entière et le **modulo**.
- Quelle est l'utilisté de cette distinction ?
- Comment calcule-t-on ces valeurs en SQL, en C++, en Java et en Python ?

2025-01-26 MCED_SQL_03a : Expression, opérateurs scalaires (v401e) © 2018-2025, M-Frs — CC BY-NC-SA 4.0
Département d'informatique, Faculté des sciences, Université de Sherbrooke, Québec

Voir https://en.wikipedia.org/wiki/Modulo

© 2018-2025, MrTs — CC BY-NC-SA 4.0