Bases de données
SQL

Opérateurs élémentaires

SQL_04
v400d

2020-09-12

Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Christina.Khnaisser@USherbrooke.ca
http://info.USherbrooke.ca/ckhnaisser
Luc.Lavoie@USherbrooke.ca
http://info.USherbrooke.ca/llavoie

Plan

- Expressions, conditions et contraintes simples
- Opérateurs prédéfinis élémentaires
  - Nombres
  - Textes
- *Les nuls de nuls pour les nuls par les nuls! [sic]*
- Exercice
- Les colles du prof

CONTRAINTES SIMPLES

- CHECK
- condition (et expression logique)
- expression

CONTRAINTES SIMPLES

- Contraintes simples :
  - Clés candidates -- BD101-SQL-LDD-01,
  - **CHECK (condition)**
- Qu'est donc une condition ?
  - Tout simplement une expression dont l'évaluation résulte en une valeur booléenne.
- Qu'est donc une expression ?
  - La dénotation d'une suite d'opérations portant sur des variables dont l'évaluation résulte en une valeur.
- Dans une contrainte simple (aussi appelée contrainte de table), les variables admises sont les attributs de la table au sein de laquelle elle est définie.

Condition (version simplifiée)

condition ::=

condition { AND | OR } condition

| NOT condition

| (condition)

| attribut IS [NOT] NULL

| expression LIKE patronL

| expression SIMILAR TO patronS

| expression { = | <> | < | <= | >= | > } expression

| expression IN listeExpressions

| expression

LE LANGAGE SQL
EXPRESSION (VERSION SIMPLIFIÉE)

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

- Une fonction est représentée par un identificateur (de fonction) suivi de ses paramètres.
- Deux catégories de fonctions
  - prédéfinies
    - *par la norme SQL*
      - les principales (mais pas la plupart) sont présentées ci-après
    - *par le SGBD*
      - se reporter à la documentation de l'éditeur du SGBD
      - faire attention à la non-transportabilité
  - définies par l'utilisateur
    - *autonomes*
    - *associées* au mécanisme de définition de type

Opérateurs et fonctions

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

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
  - reconnaissance (similar)
  - extraction (substr)
  - forme historique simplifiée (like)

Fonctions et opérateurs logiques

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

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

P	true	unknown	false
NOT P	false	unknown	true
P IS TRUE	true	false	false
P IS UNKNOWN	false	true	false
P IS FALSE	false	false	true

Fonctions et opérateurs numériques (liste partielle)

- arithmétique sur les entiers
  - +, -, *, /
  - mod (a, b)
- arithmétique sur les rationnels et les flottants
  - +, -, *, /
  - abs (x), round (x)
  - floor (x), ceiling (x)
  - sqrt (x), power (b, e)
- fonctions trigonométriques
  - sin (x), cos (x), tan (x), ...
- fonctions logarithmiques et exponentielles
  - ln (x), exp (x)
- divers
  - ...

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 !*

Fonctions et opérateurs temporels prédéfinis (liste partielle)

- current_timestamp
- current_date
- current_time

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

- extract(<field> from timestamp)
  - **<field> := YEAR | MONTH | DAY | HOUR | MINUTE | SECOND**

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 nombre réduits d'opérateurs.
- Les LR sont équivalents aux automates d'états finis.
- Leur étude sera approfondie au cours des activités
  - MAT115,
  - IFT313.
- 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).

Syntaxe des LR en SQL (1/6)

```
expR ::=
        atome
  |     expR fermeture          -- la fermeture de l'expression expR
  |     expR1 expR2             -- expR1 suivi de expR2
  |     expR1 | expR2           -- expR1 ou expR2
  |     ( expR )                -- l'expression expR elle-même
```

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
```

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

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 »

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], mais pouvant comprendre des intervalles représentés par deux caractères réunis par un tiret »

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

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.

Opérateur de conformité

- La fonction logique SIMILAR TO renvoie TRUE si et seulement si le texte de gauche est conforme à l'expression rationnelle représentée par le texte de droite :
  - texte-gauche SIMILAR TO texte-droit

Opérateur d'extraction

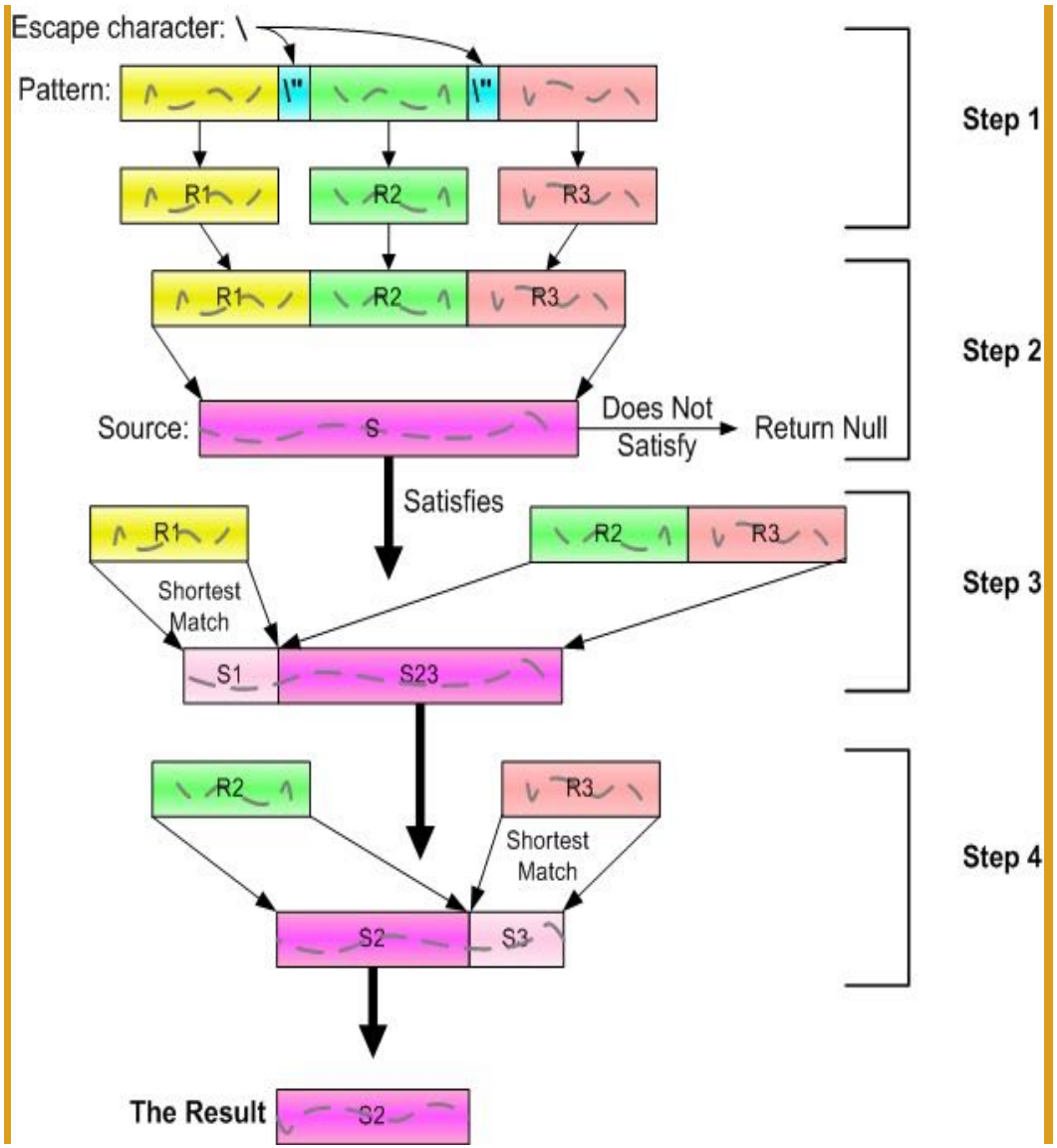
- substr (*s* similar to *p* [escape *c*])

- 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*]
- ...
- bof!

Différences entre SQL et PostgreSQL

- Le dialecte PostgreSQL met à disposition de nombreuses abréviations, comme
  - ~ pour SIMILAR TO
- et de nombreuses fonctions supplémentaires
  - regexp_replace
  - regexp_matches
  - regexp_split_to_table
  - etc.

- Voir
  - https://docs.postgresql.fr/9.6/functions-matching.html

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 SIMILAR TO.
- On trouvera une bonne introduction à la notation POSIX dans

  https://fr.wikipedia.org/wiki/Expression_rationnelle

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}'
)

CONSTRAINT gare_idGare CHECK (
  idGare SIMILAR TO 'G[0-9]{7}'
)
```

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.

NULS : MARQUEURS ET VALEURS

- Opérations
- Marqueur ou valeur ?
- Cohérence
- Annulabilité
- Conséquences

Opérations avec les nuls (1/3)

- En général, en SQL, NULL doit être considéré comme un marqueur d'attribut et non comme une valeur.
- Quelques conséquences
  - dans une expression, NULL marque l'absence de valeur
    - INSERT INTO R (a, b) VALUES (12, NULL)
    - a prend pour valeur 12; b n'a pas de valeur
  - soit a un attribut quelconque, on ne peut pas écrire
    - a = NULL
  - on doit écrire
    - a IS NULL

Opérations avec les nuls (2/3)

- Par ailleurs, soit a et b deux attributs nuls,
  - $a = b$ est inconnu
  - $a <> b$ est inconnu aussi
- En particulier,
  - $a = a$ est inconnu
  - $a <> a$ est inconnu aussi

Opérations avec les nuls (3/3)

- En général, toute évaluation d'expression nécessitant l'évaluation d'un attribut NULL entraine l'annulabilité de l'expression.
- Dans les expressions booléennes, la valeur d'un prédicat dont un des termes est NULL est inconnue (UNKNOWN).
- La logique de SQL est donc trivaluée : FALSE, TRUE, UNKNOWN.
- Les prédicats IS NUL, IS NOT NULL et les fonctions CASE, NULLIF et COALESCE font exception aux règles précédentes.

NULL : MARQUEUR OU VALEUR ?

- Notons l'ambigüité fréquente entre
  - marqueur d'attribut : c'est-à-dire une propriété d'un attribut (en extension au modèle relationnel)
  - valeur d'une expression : ce qui nécessite d'ajouter cette valeur à tous les domaines (en extension au modèle de typage).
- Malheureusement, la norme et les dialectes ont recours aux deux mécanismes!

La cohérence système logique de SQL

- Que se passe-t-il quand la condition d'une contrainte (CHECK) est UNKNOWN ?
  - Elle est réputée satisfaite!
- Que se passe-t-il quand la condition d'une restriction (WHERE) est UNKNOWN ?
  - Elle est réputée non satisfaite!!!

- Le système logique utilisé par SQL est donc incohérent!

L'annulabilité logique (unknown)

- Qu'arrive-t-il si un attribut logique (BOOLEAN) est NULL ?
- Il est considéré comme UNKNOWN.

Les conséquences

- L'égalité est incohérente en présence de NULL et UNKNOWN.
- Les opérateurs l'utilisant (dont l'union, l'intersection, la différence, la restriction, la projection et la jointure) peuvent voir des résultats incohérents.
- Conséquemment les instructions DELETE, UPDATE et SELECT peuvent également produire des résultats incohérents comme nous le verrons dans les prochains modules.
- *Il sera donc plus prudent d'éviter tout recours au NULL!*

AUTRES OPÉRATEURS

- COALESCE
- CASE
- CAST

Opérateurs COALESCE

- COALESCE (x_1, x_2, ..., x_n)
  - première(*) expression non nulle parmi x_1, x_2, ..., x_n ;
  - si toutes les expressions sont nulles, NULL.

  - (*) de gauche à droite
  - (*) dont l'indice est le plus petit

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.

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!

Opérateur CASE – « valué »

<searched case> ::=
CASE <expression> { <searched when clause> ... }
[**ELSE** <result>]
END
<searched when clause> ::=
WHEN <searched operand> **THEN** <result>
<searched operand> ::=
... la liste des valeurs du cas ...

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!

Formes abrégées du CASE

- Pour terminer, la norme précise ceci :
  - NULLIF (*V*1, *V*2) est équivalent à :
    - ```
      CASE
      WHEN V1=V2 THEN NULL
      ELSE V1
      END
      ```
  - COALESCE (*V*1, *V*2) est équivalent à :
    - ```
      CASE
      WHEN V1 IS NOT NULL THEN V1
      WHEN V2 IS NOT NULL THEN V2
      ELSE NULL
      END
      ```
  - COALESCE (*V*1, *V*2, ..., *Vn*), pour $n \geq 3$, est équivalent à :
    - ```
      CASE
      WHEN V1 IS NOT NULL THEN V1
      ELSE COALESCE (V2, ..., Vn)
      END
      ```
      ```

Opérateur CAST

- Constats :
  - Le type d'une expression n'est pas toujours univoque.
  - La valeur d'un variable peut avoir une correspondance légitime dans un type autre que celui de la variable.
- Solution :
  - L'opérateur CAST permet d'associer un type particulier à la valeur.
- Syntaxe :
  - ISO, PostgreSQL, Oracle :
    - CAST (<expr> AS <type>)
  - PostgreSQL (autre notation en sus) :
    - <expr> :: <type>

Exemples

- Évaluation
- Gaspard et Madeleine

Évaluation – Activité
Contrainte sur sigle

```
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}')
  );

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     CHAR(8) NOT NULL,
     activite      CHAR(6) NOT NULL,
     trimestre     CHAR(5) NOT NULL,
     TE            CHAR(2) 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')
      )
  );
```

Un petit exercice ?
Gaspard et Madeleine – ce n'était pas fini!

```
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_poids CHECK (0 < poids),
    CONSTRAINT Arme_noProduit CHECK
      (noProduit SIMILAR TO 'A[0-9]{5}')
);
```

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 4

Gaspard et Madeleine
Itération 4

```
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_cr0
    FOREIGN KEY (typeArme) REFERENCES DefTypeArme,
  CONSTRAINT   Arme_poids CHECK (0 < poids),
  CONSTRAINT   Arme_noProduit
    CHECK (noProduit SIMILAR TO 'A[0-9]{5}')
);
CREATE TABLE DefTypeArme (
  typeArme     VARCHAR(15) NOT NULL,
  CONSTRAINT   DefTypeArme_cc0 PRIMARY KEY (typeArme)
);
```

Gaspard et Madeleine
Est-ce fini ?

- Nous avons trouvé une solution technique, mais n'aurions-nous pas dû revenir à la modélisation au préalable ?
  - Voir le module BD010 - Gaspard et Madeleine !

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.
- La documentation de PostgreSQL (en français)
  - https://docs.postgresql.fr
- La documentation de MariaDB (en anglais)
  - https://mariadb.com/kb/en/library/documentation/
- La documentation d'Oracle (en anglais)
  - http://docs.oracle.com/cd/E11882_01/index.htm

Les colles du prof

- SQL comporte un opérateur mod (pour modulo), est-ce la même chose que le reste de division entière ?
- En mathématiques ?
- En SQL ?