

# PRÉCIS d'Algèbre Relationnelle & de SQL

## 1. OPERATEURS DE L'ALGEBRE RELATIONNEL

**ALGEBRE RELATIONNELLE** = { opérateurs sur les relations donnant en résultat des relations }

### Opérateurs ensemblistes :

#### UNION

$$T = \text{union}(R, S)$$

$$T = \{ t / t \in R \vee t \in S \}$$

L'union de 2 relations R & S, obligatoirement de même schéma, est une relation de même schéma que R & S, qui possède l'ensemble des tuples de R plus l'ensemble des tuples de S.

#### INTERSECTION

$$T = \text{intersect}(R, S)$$

$$T = \{ t / t \in R \wedge t \in S \}$$

L'intersection de 2 relations R & S, obligatoirement de même schéma, est une relation de même schéma que R & S, qui possède tous les tuples présents à la fois dans R et dans S.

#### DIFFERENCE

$$T = \text{différence}(R, S)$$

$$T = \{ t / t \in R \wedge t \notin S \}$$

La différence des 2 relations R & S, obligatoirement de même schéma, R - S, est une relation de même schéma que R & S, qui possède tous les tuples de R qui ne sont pas présents dans S.

#### PRODUIT CARTESIEN

$$T = \text{prod}(R, S)$$

$$T = \{ t_{xy} / t_x \in R \wedge t_y \in S \}$$

Le produit cartésien des 2 relations R & S, non forcément de même schéma, R x S, est une relation dont le schéma résulte de la concaténation des schémas de R & S, qui possède tous les tuples résultant de la concaténation de chacun des tuples de R avec chacun des tuples de S.

### Opérateurs relationnels :

#### PROJECTION

$$T = \text{proj}(R, a_i, a_j, \dots, a_p)$$

$$T = \text{proj}(R, A)$$

$$T = \{ t_A / \exists t_B \in R_B \wedge t_{A,B} \in R_{A,B} \}$$

La projection appliquée à la relation R, sur le sous-ensemble d'attributs A, est une relation qui a pour schéma le sous-ensemble d'attributs A de R, et qui possède l'ensemble des tuples de R réduits au sous-schéma A.

Autres notations :  $T = \Pi_{a_i, a_j, \dots, a_p}(R)$

#### RESTRICTION

$$T = \text{restrict}(R, C_r)$$

$$T = \{ t / t \in R \wedge C_r(t) \}$$

La restriction appliquée à la relation R, sur le critère de qualification  $C_r$ , est une relation qui a le même schéma que R, et qui possède le sous-ensemble des tuples de R qui satisfont au critère de qualification  $C_r$ .

Autres notations :  $T = \sigma_{C_r}(R)$

$C_r$  est une expression logique qui référence un ou plusieurs attributs de la relation R.

## JOINTURE

$T = \text{join}(R, S, C_j)$

$T = \{ t_{A,B} / t_A \in R(A) \wedge t_B \in S(B) \wedge C_j(t_{A,B}) \}$

La jointure des deux relations R & S, de schéma non forcément identique, sur le critère de jointure  $C_j$ , est une relation qui a pour schéma la concaténation des schémas de R et de S, et qui possède le sous-ensemble des tuples du produit cartésien de R et S qui satisfont au critère de qualification  $C_j$ .

Autres notations :  $T = R \bowtie_{C_j} S$

$C_j$  est une expression logique qui référence obligatoirement au moins un attribut de R et au moins un attribut de S.

**Equijointure** :  $C_j$  est de la forme  $R.\text{attribut}_i = S.\text{attribut}_j$

**Inéquijointure** :  $C_j$  est de la forme  $R.\text{attribut}_i \langle \text{opérateur} \rangle S.\text{attribut}_j$   
où opérateurs  $\in \{<, >, <=, >=, <>\}$

**Jointure naturelle** : jointure entre deux relations ayant un attribut de même nom et qui contient les tuples concaténés des deux relations qui ont la même valeur pour l'attribut commun.

**Jointure externe** : jointure de deux relations R & S, complétée des tuples de R et de S non inclus dans la jointure avec la valeur NULL pour les attributs de l'autre relation.

**Semi-jointure** : jointure entre deux relations R & S qui a pour résultat les tuples de R qui appartiennent à la jointure de R & S.

## DIVISION

$T = \text{div}(R, S)$

où  $R(a_1, a_2, \dots, a_p, a_{p+1}, \dots, a_n)$ ,  $S(a_{p+1}, \dots, a_n)$ , et  $T(a_1, a_2, \dots, a_p)$

$T = \{ t_{a_1, a_2, \dots, a_p} / \forall t_{a_{p+1}, \dots, a_n} \in S, t_{a_1, a_2, \dots, a_p, a_{p+1}, \dots, a_n} \in R \}$

La division de la relation R par la relation S dont le schéma correspond au sous-schéma 'droit' de R, est une relation qui a pour schéma le sous-schéma complémentaire du schéma de S dans R, et qui possède le sous-ensemble des tuples de R réduit à ce sous-schéma qui, concaténés à n'importe quel tuple de S, donnent un tuple de R.

La division permet de répondre aux interrogations de type " $\forall x$ , trouver y".

## 2. CONCEPTS ADDITIONNELS : FONCTIONS DE CALCUL, AGREGATS

### Expressions d'attributs

expressions arithmétiques construites à partir d'attributs de relation et de constantes par application de fonctions arithmétiques successives.

Ces expressions peuvent être utilisées comme arguments des opérateurs de Projection, Restriction, ou Jointure.

Exemples :  $(\text{pilote.salaire} * 1,05)$   
 $(\text{commande.quantite} * \text{produit.prix\_unitaire})$

## Fonctions de calcul sur ensemble

SOMME (<expression>)

SUM (<expression>)

<expression> = <attribut> | <expression d'attribut>

Effectue la somme des valeurs de l'expression pour un ensemble de tuples.

MOYENNE (<expression>)

AVG (<expression>)

<expression> = <attribut> | <expression d'attribut>

Effectue la moyenne des valeurs de l'expression pour un ensemble de tuples.

COMPTE (<expression>)

COUNT (<expression>)

<expression> = <attribut> | \*

Compte le nombre de tuples de l'ensemble. Peut s'utiliser avec « DISTINCT » pour compter le nombre de valeurs distinctes de l'expression dans l'ensemble de tuples.

MINIMUM (<expression>)

MIN (<expression>)

<expression> = <attribut> | <expression d'attribut>

Sélectionne l'élément ayant la plus petite valeur de l'expression dans l'ensemble de tuples.

MAXIMUM (<expression>)

MAX (<expression>)

<expression> = <attribut> | <expression d'attribut>

Sélectionne l'élément ayant la plus grande valeur de l'expression dans l'ensemble de tuples.

**Une fonction de calcul sur ensemble peut être utilisée comme argument de l'opérateur Projection. Dans ce cas, l'argument ou les arguments de la Projection doivent être uniquement des fonctions :**  
**proj(R, fonction(expression) [,fonction(expression)[, ...]]).**

## Opération d'agrégat

Agrégat(R, {a<sub>1</sub>, a<sub>2</sub>, ..., a<sub>n</sub>}, F<sub>i</sub>(b<sub>1</sub>, b<sub>2</sub>, ..., b<sub>n</sub>)[F<sub>j</sub>(...), ...])

où R est une relation,

a<sub>i</sub> un attribut de R qualifié d'attribut de regroupement

F<sub>i</sub>(b<sub>i</sub>) une fonction de calcul sur ensemble appliquée aux attributs b<sub>i</sub>

L'opérateur "Agrégat" effectue un partitionnement horizontal de la relation R en regroupant ses tuples par valeurs identiques d'un attribut ou d'un groupe d'attributs (a<sub>i</sub> : attributs de regroupement), puis applique une ou plusieurs fonctions de calcul sur ensemble (F<sub>i</sub>) à chaque partition obtenue, et termine par une projection sur l'ensemble d'attributs de regroupement et les fonctions de calcul.

### 3. LANGAGE DE MANIPULATION DES DONNEES SQL : SOUS-LANGAGE D'INTERROGATION

#### *Un ordre unique : SELECT*

#### **SELECT [DISTINCT]**

<liste\_d'attributs\_projetés> | \*|<expression\_d\_attribut>  
 | <fonction(attribut | <expression\_d\_attribut)>  
 | <constante littérale>  
 | <expression\_de\_table>

**FROM** < nom\_de\_table> | <liste de nom\_de\_table> | < jointure\_de\_table>

**[WHERE <critère\_de\_qualification>]**

**[GROUP BY < reference\_attribut> [, <reference\_attribut> [, ...]]**

**[HAVING <critère\_de\_qualification>] ]**

**[ORDER BY < reference\_attribut> [, <reference\_attribut> [, ...]] [ASC|DESC] ] > ;**

<liste\_d'attributs\_projetés> ::= <reference\_attribut> [,<reference\_attribut>  
 [,<reference\_attribut>[, ...]]]

<expression\_d\_attribut> ::=

<reference\_attribut> <opérateur\_arithmétique> <reference\_attribut>

| <reference\_attribut> <opérateur\_arithmétique> <valeur>

| <fonction> <opérateur\_arithmétique> <fonction>

| <expression\_d\_attribut> <opérateur\_arithmétique> <expression\_d\_attribut>

<fonction(reference\_attribut | <expression\_d\_attribut)> >:

fonctions de calcul sur ensemble ou fonction de la bibliothèque du SGBDR

<expression\_de\_table> ::=

<ordre SELECT imbriqué dans la clause SELECT >

<reference\_attribut> ::= <nom\_table> . <nom\_attribut> | <nom\_attribut>

<liste de nom\_de\_table> ::=

<nom\_table> , <nom\_table> [, <nom\_table> [, ...]]

< jointure\_de\_table> ::=

<nom\_table> **join** <nom\_table> **on** <critere\_de\_jointure>

| <nom\_table> **join** (<jointure\_de\_table>) **on** <critere\_de\_jointure>

| (<jointure\_de\_table>) **join** <nom\_table> **on** <critere\_de\_jointure>

| (<jointure\_de\_table>) **join** (<jointure\_de\_table>) **on** <critere\_de\_jointure>

Dans les définitions ci-dessus, < nom\_table> peut être remplacé par

<expression\_de\_table> où

<expression\_de\_table> ::=

<ordre SELECT imbriqué dans la clause FROM >

**Critère de qualification**

C'est une expression logique référençant un ou plusieurs attributs, utilisant les opérateurs de comparaison classiques (=, <, >, <=, >=), ainsi que les opérateurs logiques AND, OR, NOT, et également des opérateurs spécifiques du langage SQL.

Parmi ces derniers, on peut citer :

**BETWEEN**

syntaxe d'utilisation : <nom\_attrib> BETWEEN <valeur<sub>1</sub>> AND <valeur<sub>2</sub>>

Cet opérateur est équivalent à l'expression classique:

(<nom\_attrib> >= <valeur<sub>1</sub>>) AND (<nom\_attrib> <= <valeur<sub>2</sub>>)

<valeur<sub>1</sub>> et <valeur<sub>2</sub>> peuvent être le résultat de SELECT imbriqué (Cf. ci-dessous)

**IN**

syntaxe d'utilisation : <nom\_attrib> IN (<valeur<sub>1</sub>>, <valeur<sub>2</sub>>, ..., <valeur<sub>n</sub>>)

Cet opérateur est équivalent à l'expression classique:

(<nom\_attrib> = <valeur<sub>1</sub>>) OR (<nom\_attrib> = <valeur<sub>2</sub>>) . . . OR (<nom\_attrib> = <valeur<sub>n</sub>>)

**LIKE**

syntaxe d'utilisation : <nom\_attrib> LIKE <chaîne de caractères générique>

Cet opérateur permet de comparer la valeur d'un attribut de type chaîne de caractères à une chaîne générique.

Les caractères génériques (joker) définis en standard sont:

**%** pour désigner une séquence de n caractères quelconques, avec n>=0

**\_** pour désigner un seul caractère quelconque

**IS NULL**

syntaxe d'utilisation : <nom\_attrib> IS NULL

Cet opérateur permet de savoir si un attribut a été ou non affecté d'une valeur

**EXISTS**

syntaxe d'utilisation : EXISTS <sous-requête d'interrogation>

Cet opérateur permet de savoir si la sous-requête ramène un résultat, c'est-à-dire au moins un tuple, et dans ce cas l'expression logique est vraie. Si la sous-requête ne ramène aucun résultat, l'expression logique est fausse.

**Clause DISTINCT**

Permet de supprimer les doublons dans le résultat d'un ordre SELECT.

**Clause ORDER BY**

Permet de trier le résultat d'un ordre SELECT.

**Clause GROUP BY**

C'est la clause qui traduit l'**agrégat** en SQL. D'où la contrainte suivante :

la liste d'attributs projetés à l'exclusion de toute fonction de calcul, doit être identique à la liste d'attributs de regroupement; i.e liste des attributs de la clause SELECT à l'exclusion de toute fonction de calcul = liste des attributs de la clause GROUP BY.

**Clause HAVING**

C'est un critère de qualification applicable au résultat de l'agrégat. Cette clause n'est présente qu'à la suite d'une clause GROUP BY. Elle référence les attributs de regroupement et/ou les fonctions de calcul utilisées dans l'agrégat.

En dehors de cette spécificité, la définition du critère de qualification de la clause HAVING est identique à celle du critère de qualification dans la clause WHERE.

**Facilités d'écriture**

➤ Renommer les colonnes du résultat  
usage : <nom\_attribut> **as** <libellé\_colonne>

➤ Donner un **alias** aux tables

usage : <nom\_table> <alias>

où <alias> est une chaîne courte de caractères (entre 1 et 3 caractères généralement) qui est séparé du nom de table par au moins un espace

l'alias sert d'une part à simplifier l'écriture de l'ordre sql, d'autre part à identifier une occurrence particulière d'une table référencée plusieurs fois dans la même requête

**Traduction des opérations UNION, INTERSECTION, DIFFERENCE**

La syntaxe est similaire pour les trois opérateurs

<ordre select> UNION <ordre select>

<ordre select> INTERSECT <ordre select>

<ordre select> EXCEPT <ordre select>

**Select imbriquées**

C'est un ordre SQL dont le résultat est utilisé comme valeur de comparaison dans le critère de qualification de la clause WHERE ou la clause HAVING d'un ordre SQL englobant.

Plusieurs niveaux d'imbrication sont possibles, et théoriquement infini.

Le « select imbriqué », selon l'usage qui en est fait, doit impérativement ramener une seule valeur, ou peut ramener plusieurs valeurs :

```
SELECT .....
FROM .....
WHERE AttributX = (SELECT ..... )
..... ;
```

*Dans ce cas le SELECT ramène IMPÉRATIVEMENT une seule valeur*

```
SELECT .....
FROM .....
WHERE AttributX IN (SELECT ..... ) ;
```

*Dans ce cas le SELECT peut ramener plusieurs valeurs*

Les opérateurs suivants sont utilisables :

- opérateurs classiques : =, <, >, <=, >=, <>

- IN / NOT IN

- <op> ANY

où <op> ∈ { =, <, >, <=, >=, <> }

est vrai si <attribut> <op> <valeur> est vrai pour au moins une valeur du résultat du select imbriqué

- <op> ALL

où <op> ∈ { <, >, <=, >=, <> }

est vrai si <attribut> <op> <valeur> est vrai pour toutes les valeurs du résultat du select imbriqué

- EXISTS

syntaxe : exists (select ... )

est vrai si le résultat du select imbriqué n'est pas vide, i.e contient au moins un tuple

**La DIVISION en SQL**

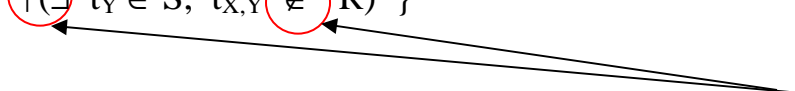
Rappel de la définition :  $T : \text{div}(R,S)$  avec  $R(X,Y)$  et  $S(Y)$

$$T = \{ t_X / \forall t_Y \in S, t_{X,Y} \in R \}$$

Pour traduire la division en SQL, l'équivalence logique suivante est appliquée :

$$\forall x, P(x) \equiv \neg (\exists x, \neg P(x))$$

Ce qui permet de redéfinir la division comme suit

$$T = \{ t_X / \neg (\exists t_Y \in S, t_{X,Y} \notin R) \}$$


D'où la traduction en SQL de la division par l'usage de deux select imbriqués avec l'opérateur **NOT EXISTS**

**Expression de table**

Ce peut être un Select imbriquée dans la clause SELECT

SELECT , att1, attj, ..., (select .....)

FROM .....

Où le select doit ramener une valeur unique

Ce peut être un Select imbriquée dans la clause FROM

dans une liste de table :

SELECT , att1, attj, ...

FROM tab1, (select ..... ) tabj, ....

dans une jointure de table :

SELECT , att1, attj, ...

FROM tab1 JOIN (select ..... ) tabj ON .....

Où tabj est un alias à spécifier obligatoirement

## **4. LANGAGE DE MANIPULATION DES DONNEES SQL : SOUS-LANGAGE DE MISE A JOUR**

➤ Insertion des données : **INSERT**

2 formats possibles :

INSERT INTO < nom\_de\_table > [attribut1, attribut2, ... , attributi, ...]

VALUES ( <expression [ , <expression> ... ] > ;

ou

INSERT INTO < nom\_de\_table > [attribut1, attribut2, ... , attributi, ...]

< ordre\_select > ;

➤ Suppression des données : **DELETE**

DELETE FROM < nom\_de\_table >

[WHERE < critère\_de\_qualification > ] ;

➤ Mise à jour des données : **UPDATE**

UPDATE < nom\_de\_table >

SET < nom\_d'attribut\_1 > = < expression1 | (ordre\_select) >

[ , < nom\_d'attribut\_2 > = < expression2 | (ordre\_select) > ... ]

[WHERE < critère\_de\_qualification > ] ;