Éléments d'algèbre relationnelle

I. Introduction

A. Origine

L'algèbre relationnelle a été introduite dans les années 1970 par Edgar Frank Codd (*« Ted » Codd, 1923-2003*), alors informaticien chez IBM, pour formaliser les opérations sur les relations, à partir de la théorie des ensembles. L'objectif était de créer une plus grande indépendance entre programmes et représentation internes des données et disposer de règles permettant de résoudre les problèmes de cohérence et de redondance de données dans les fichiers.

Ses travaux sont à l'origine des bases de données relationnelles, modèle de BD le plus utilisé encore aujourd'hui, et de SQL, langage déclaratif de manipulation des données d'une BD relationnelle, à partir des opérateurs de l'algèbre relationnelle.

Le modèle relationnel s'inscrit également dans les phases de la conception des bases de données.

Cf.Article de EF Codd : http://www.databaseanswers.org/downloads/Codds_1970_Paper.pdf

B. Les relations

1. Terminologie

La relation (mathématique) entre 2 ensembles (binaire) E1 et E2 est formée par le sous-ensemble du produit cartésien E1 X E2. Chacun des 2 ensembles est défini par son domaine de valeurs, ensemble des valeurs pouvant être prises par ses éléments. La relation forme ainsi un ensemble de couples, chacun étant formé d'un 1^er^ élément appartenant à E1 et d'un 2^nd^ appartenant à E2.

Le terme n-uplets (tuples en anglais), qualifie l'ensemble de « couples » formés par le produit cartésien de n ensembles.

Un **domaine** (- *de valeurs*) est un ensemble de valeurs. Un domaine possède un identificateur.

Exemple de domaines :

- les nombre entiers de 1900 à 1999 : {1900, 1901, 1902,..., 1998, 1999} ➔ annee
- les nombre entiers de 1 à 31 : {1, 2, 3,..., 30, 31} ➔ jour
- les jours de la semaine : {dimanche, lundi, mardi, ..., vendredi, samedi} ➔ jourSemaine
- les mois de l'année : {janvier, février, mars, ..., novembre, décembre} ➔ mois

Une **relation** est un sous-ensemble du produit cartésien d'une liste de domaines. Une relation possède un identificateur.

***Rappel** : Le produit cartésien de deux ensembles X et Y est l'ensemble de tous les couples, dont la première composante appartient à X et la seconde à Y. L'ensemble ainsi produit peut former un produit cartésien avec un 3^ème^ ensemble Z, etc.*

Exemple d'une relation « dateNaissanceEtudiant » :

- Le produit cartésien **jourSemaine X jour X mois X annee** constitue un ensemble composé de toutes les combinaisons possibles (certaines n'existeront jamais, par exemple le 30 février)
- la relation « dateNaissanceEtudiant » sera formée par un sous-ensemble de ce produit cartésien (en effet, certains jours aucun étudiant ne sera né)

Un **attribut** est ainsi un des éléments d'une relation. Chaque attribut appartient à un domaine de valeur et possède un identificateur.

Exemples d'attributs :

- jourNaissance appartenant au domaine jour
- moisNaissance appartenant au domaine mois

Ainsi, un **schéma de relation** est décrit par un **identificateur de relation** et un **ensemble d'attributs**, chacun appartenant à un domaine de valeurs différent.

Le schéma de la relation est ainsi :

DateNaissanceEtudiant (jourSemNaissance : jourSemaine**, jourNaissance** : jour**, moisNaissance** : mois**, anneeNaissance** : annee**)**

Le **degré** d'une relation correspond à son nombre d'attributs.

Le degré de la relation « DateNaissanceEtudiant » est 4.

et être formée par le sous-ensemble suivant :

{jeudi, 16, mai, 1985}, {lundi, 29, juin, 1987}, {vendredi, 6, octobre, 1989}, {mercredi, 16, juin, 1993}

La **cardinalité** de la relation correspond au nombre de tuples de la relation.

La cardinalité de la relation « DateNaissanceEtudiant » est 4.

Une **occurrence** d'une relation désigne l'un des tuples de la relation.

Une occurrence de la relation est **{jeudi, 16, mai, 1985}**.

2. Exemples de relation

On considère la relation R définie par les attributs A, B et C :

- en compréhension (*ou en intention*) :
  - **R(A** : caractere**, B** : caractere**, C** : caractere**)**
  - Degree de la relation : 3
- en extension :
  - R = { {a,b,c}, {d,a,f}, {c,b,d} }
  - Cardinalité de la relation 3

  Ou bien sous forme tabulaire (= de tableau) :

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

3. Exemple de relation

On considère la relation « Individu » définie par les attributs « numero », « nom »,« prénom » et « dateNaissance » :

- en compréhension :
  - **Individu (numero** : entier**, nom** : chaine**, prenom** : chaine**, dateNaissance** : date**)**
  - degré de la relation : 4
- en extension :
  - **Individu = { { 1, Dupont, Jacques, 15/01/1920 },**
    { 2, Durand, Pierre, 20/07/1949}, { 3, Lambert, Paul, 03/12/1974 } }

  - cardinalité de la relation : 3

Ou bien sous forme tabulaire :

Individu	Numero	Nom	Prenom	DateNaissance
	1	Dupont	Jacques	15/01/1920
	2	Durand	Pierre	20/07/1949
	3	Lambert	Paul	03/12/1974

C. Intégrité des relations

La notion de clef permet de maintenir l'intégrité des valeurs dans les relations :

- CLEF **PRIMAIRE** (anglais : Primary Key): c'est la clef qui assure l'unicité d'un tuple d'une relation ; tous les autres attributs d'une relation sont en dépendance unique de la clef primaire ; la clef peut être simple ou composite ;
  - on la représente soulignée «numP»
- CLE **CANDIDATE** : c'est une colonne qui n'est pas clef primaire, mais qui aurait pu l'être et dont on souhaite, parfois, garantir l'unicité des valeurs (un code INSEE, une adresse mail);
  - On la représente : «immatAv» ;
- CLEF **ETRANGERE** (anglais : Foreign Key): un attribut clef étrangère dans une relation est clef primaire dans une autre table ; elle matérialise un lien logique entre ces 2 relation ;
  - on la représente généralement préfixée d'un « # » : «#numAv»

Exemples 1 :

- Pilote (numP, villeP)
- Avion (numAv, nomAv, capaAv, locAv, immatAv)
- Vol (numVol, villeDep, villeArr, #numP, #numAv)
  - #numP fait reference à numP dans Pilote
  - #numAv fait reference à NumAv dans Avion

Exemple 2 :

- Ouvrage (num_ouvrage, titre, annee, #reliure)
  - #reliure fait référence à reliure dans Reliure
- Reliure (reliure, prix)

D. • L'algèbre relationnelle

L'algèbre relationnelle correspond aux opérations qu'on applique à des relations. Le résultat produit par l'application d'un opérateur de l'algèbre relationnelle à une relation est une nouvelle relation.

L'algèbre relationnelle comporte deux familles d'opérateurs :

- **Les opérateurs unaires** (mise en jeu d'une seule relation) : **PROJECTION** et **SELECTION**
- **Les opérateurs ensemblistes** (mise en jeu de 2 relations): **UNION**, **DIFFERENCE** et **PRODUIT CARTESIEN**

Ces 5 opérations forment un ensemble cohérent et minimal. Des extensions à ces opérations de bases permettent d'en améliorer l'efficacité à travers une simplification de leur écriture (**INTERSECTION**, **JOINTURE**,...) ou d'effectuer des calculs (**COMPTE**, **SOMME**,…).

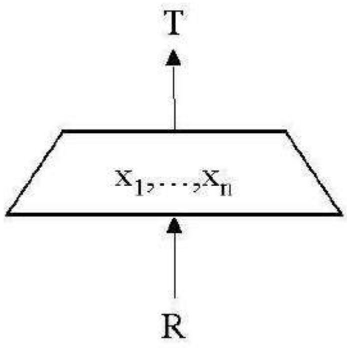
Remarque : le symbolisme de représentation des opérateurs peut varier d'un auteur à l'autre. Vous trouvez les exemples de représentations courants.

L'algèbre relationnelle est un langage théorique d'interrogation des relations et forme la base des langages d'interrogation de base de données relationnelle (SQL, par exemple).

II. LES OPERATIONS UNAIRES

Les opérations unaires d'appliquent à une seule relation et ont pour résultat une nouvelle relation.

A. PROJECTION



La projection est une opération de **RESTRICTION** sur les **COLONNES**.

La projection d'une relation R consiste en la mise en place d'une nouvelle relation en ne retenant que certains attributs et en supprimant les tuples dupliqués.

Notation :

$$T = \pi_{xi,..,xn} (R)$$

ou

$$T = \textbf{PROJ}\text{ECTION} (R , x_1, \ldots, x_n)$$

On utilise parfois le caractère « * » pour conserver toutes les attributs de la relation d'origine ou bien la relation résultante est égale à la relation initiale : T = R (peu d'intérêt)

R

A	B	C
a	b	c
d	a	f
c	b	d

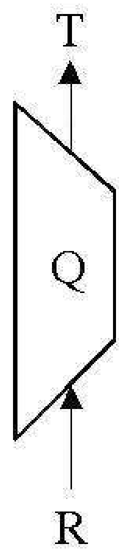
T = PROJECTION(R, A, B)

CHOIX D'UN SOUS-ENSEMBLE DE COLONNES DE LA RELATION DE DEPART

T

A	B
a	b
d	a
c	b

B. SELECTION



La sélection est une opération de **RESTRICTION** sur les **LIGNES**.

La sélection consiste à extraire de la relation R considérée un sous-ensemble de tuples satisfaisant à certains critères.

On appelle ces critères ‘ qualification’ (en abrégé Q).

La qualification Q peut être exprimée à l'aide de constantes, d’opérateurs relationnels (>, <, >=, <=, = , <>) et d’opérateurs logiques (OU, ET , NON).

Notation :

$T = \sigma_Q (R)$

ou

T = **SELECT**ION (R , Q)

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

T = SELECTION (R, A<>’a’)

CHOIX D’UN SOUS-ENSEMBLE DE TUPLES EN FONCTION D’UN CRITERE DE SELECTION (qualification)

T	A	B	C
	d	a	f
	c	b	d

C. Expressions algébriques

Afin de répondre à des demandes plus complexes, il est possible combiner les opérateurs pour former des expressions permettant de répondre à toutes formes de questions.

Ainsi, pour obtenir, à partir de la relation R, les colonnes A et B de l'ensemble des tuples pour lesquelles la valeur de l'attribut A est différente de 'a', nous pouvons écrire :

R1 = SELECTION (R , A<>'a')

R1	**A**	**B**	**C**
	d	a	f
	c	b	d

Puis :

T = PROJECTION(R1 , A, B)

T	**A**	**B**
	d	a
	c	b

Soit : **T = PROJECTION (SELECTION (R , A<>'a') , A, B)**

III. LES OPERATIONS ENSEMBLISTES

Les opérations ensemblistes s’appliquent à 2 relations pour former une nouvelle relation.

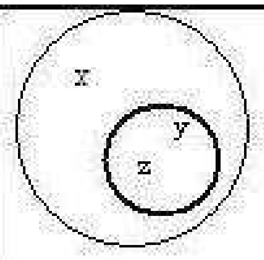
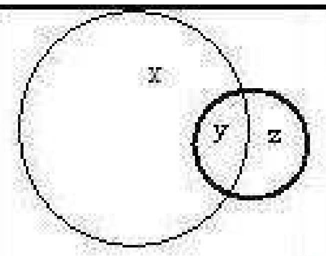
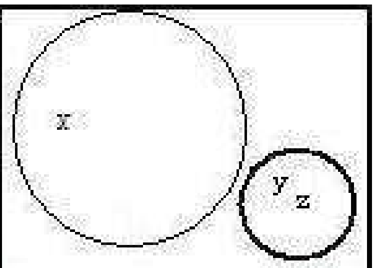
Les opérations **UNION** et **DIFFERENCE** exigent des relations qu’elles soient « **union compatibles** », c'est à dire qu'elles aient le même schéma :

- même nombre d’attributs (même degré)
- les attributs associés 2 à 2 sont issus de mêmes domaines

Remarque : les doublons sont éliminés. (*tuples pour lesquels les valeurs de tous les attributs ont la même valeur*)

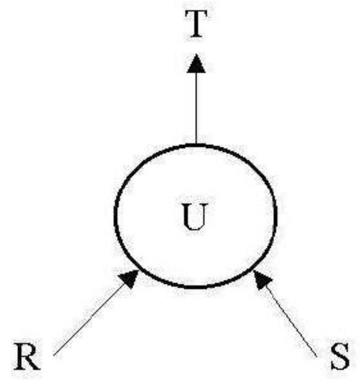
A. Opérations sur les ensembles - rappels

Soient deux ensembles R et S (*S est en gras sur les schémas*). Pour les 3 cas présentés, vous trouverez le résultat en extension et la cardinalité (nombre d’éléments) de l'ensemble résultant, T.

3 exemples d'ensembles	 R = { x,y,z } Card(R)=3 S = { y, z } Card(S)=2	 R = { x, y } Card(R)=2 S = { y, z } Card(S)= 2	 R = { x } Card(R)=1 S = { y, z } Card(S)= 2
INTERSECTION **T = R ∩ S**	T = { y, z } Card(T)=2	T = { y } Card(T)=1	T = { vide } Card(T)=0
UNION **T = R ∪ S**	T = { x, y, z } Card(T)=3	T = { x, y, z } Card(T)=3	T = { x, y, z } Card(T)=3
DIFFERENCE **T = R − S** **T = S − R**	T = { x } Card(T)=1 T = { vide } Card(T)=0	T = { x } Card(T)=1 T = { z } Card(T)=1	T = { x } Card(T)= 1 T = { y, z } Card(T)= 2

- T = R U S ➔ card(R) + card(S) - card(R ∩ S)
- T = R - S ➔ card(R) - card(R ∩ S)
- T = S - R ➔ card(S) - card(R ∩ S)

B. UNION

	L’**union** de 2 relations R et S de même schéma est une relation T de même schéma contenant l’ensemble des tuples appartenant soit à R, soit à S, soit à la fois à R et S Notations : T = (R U S) Ou T=**UNION**(R, S)

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

Les relations sont union-compatibles

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

Les doublons sont supprimés

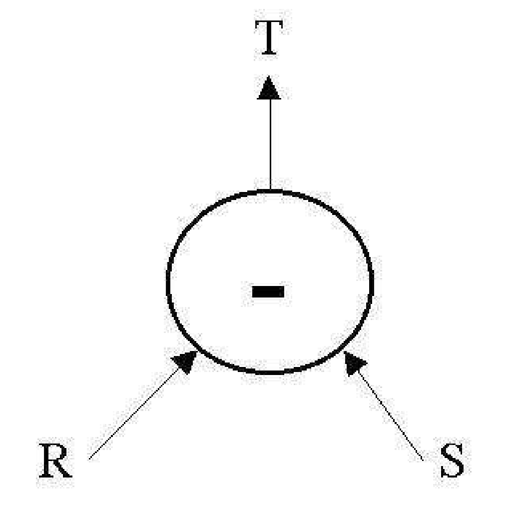
TOUS LES TUPLES DES DEUX RELATIONS EN SUPPRIMANT LES DOUBLONS

T	A	B	C
	a	b	c
	d	a	f
	c	b	d
	b	g	a

Provient de R (lignes a b c, d a f, c b d)

Provient de S (ligne b g a)

C. DIFFERENCE



La **différence** entre 2 relations R et S de même schéma dans l'ordre (R - S) est la relation T de même schéma contenant les tuples appartenant à R sauf ceux qui appartiennent aussi à S.

Notation :

T= (R - S)

Ou

T= **MINUS**(R,S)

T = DIFFERENCE(R, S)

Les relations sont union-compatibles

R

A	B	C
a	b	c
d	a	f
c	b	d

S

A	B	C
b	g	a
d	a	f

Les tuples présents dans S ne sont pas repris

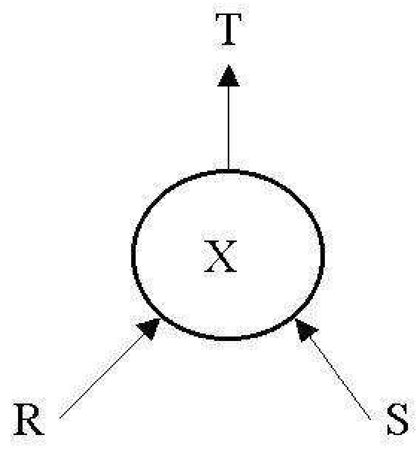
On enlève de R ce qui est aussi dans S (TOUTES LES LIGNES DE LA PREMIERE RELATION A L'EXCEPTION DE CELLES DE LA DEUXIEME)

T

A	B	C
a	b	c
c	b	d

Les tuples de R qui ne sont pas dans S

D. PRODUIT CARTESIEN



Le produit cartésien de 2 relations R et S de schéma quelconque est une relation T ayant pour attributs les attributs de R et de S, et dont les tuples sont constitués par la combinaison de chaque tuple de R avec chacun des tuples de S,.

Le produit cartésien permet la construction de toutes les combinaisons possibles entre les tuples de 2 relations.

Notation :

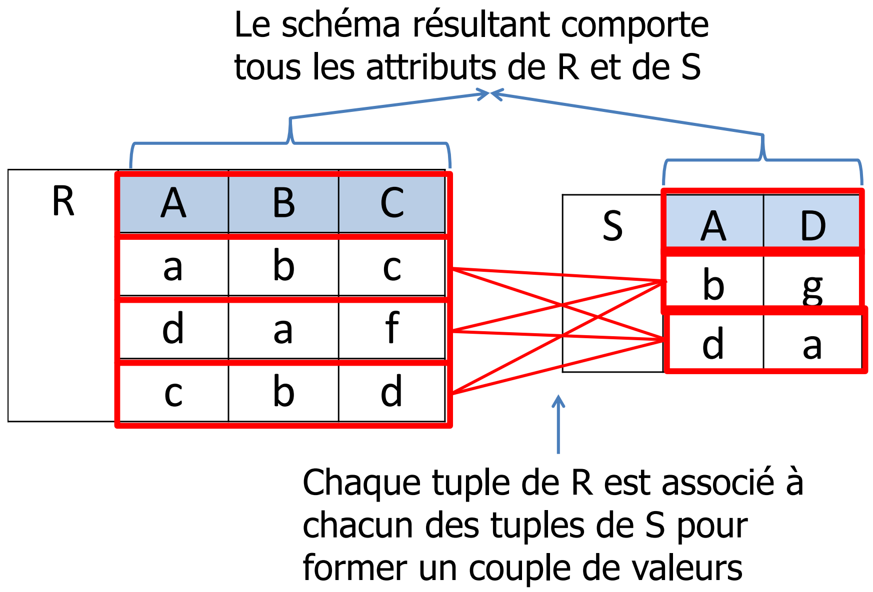
T = (R X S)

ou

T=**PRODU**IT(R, S)

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	D
	b	g
	d	a



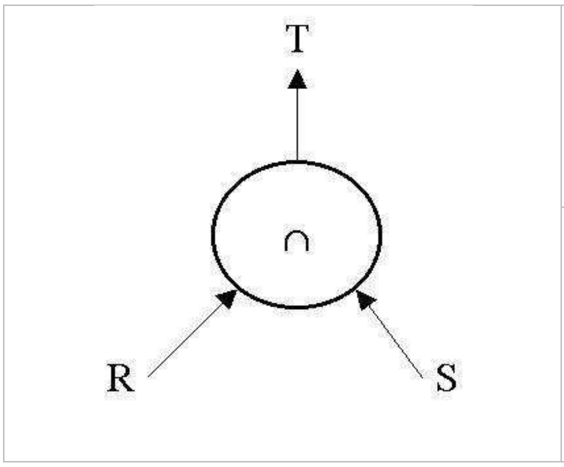
T	R.A	R.B	R.C	S.A	S.D
	a	b	c	b	g
	a	b	c	d	a
	d	a	f	b	g
	d	a	f	d	a
	c	b	d	b	g
	c	b	d	d	a

Les tuples de R Les tuples de S

IV. Les opérations dérivées

Elles peuvent être construites à partir des opérations de base.

A. INTERSECTION



L'intersection de 2 relations R et S de même schéma est une relation T de même schéma contenant les tuples appartenant à la fois à R et à S.

Notation :

$T = (R \cap S)$

ou

T = **INTER**SECT(R, S)

L'intersection peut être construite à partir des opérateurs de base :

$(R \cap S) = R - (R - S) \ = S - (S - R)$

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

Les relations sont union-compatibles

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	B	C
	b	g	a
	d	a	f

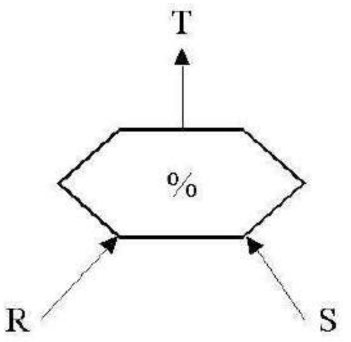
Les tuples présents dans R et dans S sont repris

LES LIGNES QUI SONT A LA FOIS DANS LA PREMIERE ET DANS LA DEUXIEME RELATION

T	A	B	C
	d	a	f

Tuples communs à R et à S

B. QUOTIENT ou DIVISION



Le quotient (ou division) : de la relation R de schéma R (A1, A2,...An) par la sous-relation S de schéma S (Ap+1, ...An) est la relation T de schéma T(A1, A2,...Ap) formée de tous les tuples qui, combinés à chaque tuple de S, donnent toujours un tuple de R.

Notation :

T = (R / S)

ou

T = **DIV**ISION (R, S)

T = QUOTIENT (R, S)

Le quotient permet de rechercher l'ensemble de tous les sous-tuples d'une relation satisfaisant une sous-relation décrite par l'opération diviseur.

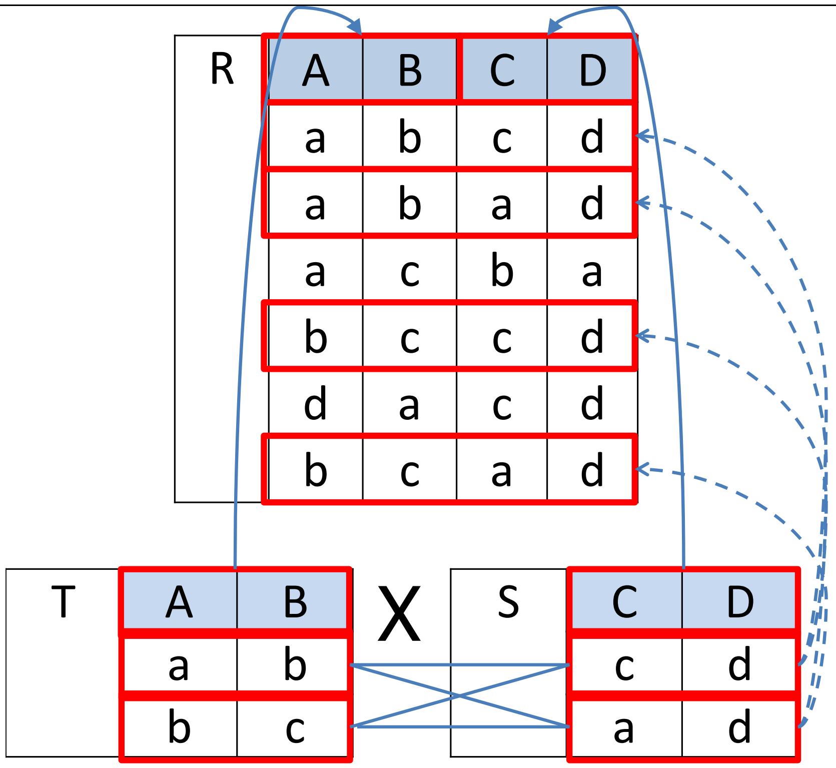
R	A	B	C	D
	a	b	c	d
	a	b	a	d
	a	c	b	a
	b	c	c	d
	d	a	c	d
	b	c	a	d

S	C	D
	c	d
	a	d

R	A	B	C	D
	a	b	c	d
	a	b	a	d
	a	c	b	a
	b	c	c	d
	d	a	c	d
	b	c	a	d

S	C	D
	c	d
	a	d

T	A	B
	a	b
	b	c



Le produit cartésien de T et S va toujours donner un tuple de R

Exemple d'utilisation : rechercher tous les produits qui existent dans une gamme de couleurs(ou une gamme de prix), ou bien tous les livres qu'on trouvera dans une gamme de reliures, ou bien les coureurs qui ont participé à une liste de grands prix,etc.

C. JOINTURES

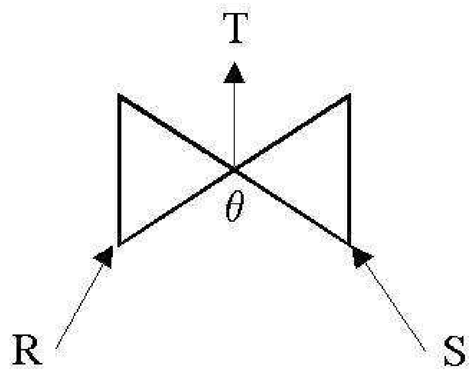
L’opération de base de la jointure est le produit cartésien.

1. Jointure interne (thêta-jointure)

Cette opération est essentielle dans les systèmes relationnels pour rassembler de manière cohérente des attributs provenant de plusieurs relations.

La jointure interne, ou thêta-jointure (INNER JOIN) est une produit cartésien de R et S qui conserve seulement les tuples satisfaisant une qualification (qualification = comparaison entre 2 attributs) :

- L'**équijointure** est une jointure ayant pour qualification l'égalité entre 2 attributs;
- La **non-équijointure** est une jointure ayant pour qualification un opérateur autre que l'égalité entre 2 attributs



La jointure de 2 relations R et S selon une qualification Q est l'ensemble des tuples du produit cartésien R X S satisfaisant la qualification Q. *(voir l’opérateur Selection)*.

La jointure exige que les 2 relations aient un domaine en commun.

La jointure est équivalente à l’application d’une sélection à un produit cartésien.

Notation :

$$T = (R \rhd\lhd_{Q})S$$

ou

T = **JOIN**TURE(R, S , Q)

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	D
	b	g
	d	a

T = JOIN (R, S, R.B = S.A)

R X S	R.A	R.B	R.C	S.A	S.D
	a	b	c	b	g
	a	b	c	d	a
	d	a	f	b	g
	d	a	f	d	a
	c	b	d	b	g
	c	b	d	d	a

Q
R.B = S.A

T	R.A	R.B	R.C	S.A	S.D
	a	b	c	b	g
	c	b	d	b	g

2. Jointure naturelle

La **jointure naturelle** (NATURAL JOIN) est une jointure de R et S sur tous les attributs de même nom, suivi de la projection qui permet de supprimer les attributs répétés.

Attention :

- si aucun nom d'attribut n'est commun, on obtient un produit cartésien
- si des attributs portent le même nom sans toutefois avoir le même sens, on obtiendra une jointure incohérente...

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	D
	b	g
	d	a

T = NATURAL JOIN (R, S)

R X S	R.A	R.B	R.C	S.A	S.D
	a	b	c	b	g
	a	b	c	d	a
	d	a	f	b	g
	d	a	f	d	a
	c	b	d	b	g
	c	b	d	d	a

T	A	B	C	S.A	D
	d	a	f	d	a

3. La jointure externe

La **jointure externe** (OUTER JOIN) est une jointure de R et S qui conserve tous les tuples, de l'une ou l'autre des relations et les tuples issus de la jointure des 2 relations :
La jointure de R et S conserve tous les tuples :

- de R pour une **jointure externe gauche**, (LEFT OUTER JOIN)
- et de S pour une **jointure externe droite** (RIGHT OUTER JOIN)
- ou des 2 relations pour une **jointure totale** (FULL OUTER JOIN).

➔Les valeurs des tuples ne satisfaisant pas à la qualification ont pour valeur 'NULL'.

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

S	A	D
	b	g
	d	a

T = LEFT OUTER JOIN (R, S, R.A = S.A)

Ou

T = LEX JOIN(R, S, R.A = S.A)

(REX_JOIN pour une jointure externe droite)

(FEX_JOIN pour une jointure externe complète)

On conserve tous les tuples de la table à gauche de la jointure

R X S	R.A	R.B	R.C	S.A	S.D
	a	b	c	b	g
	a	b	c	d	a
	d	a	f	b	g
	d	a	f	d	a
	c	b	d	b	g
	c	b	d	d	a

Q
R.A = S.A

T	R.A	R.B	R.C	S.A	S.D
	a	b	c	null	null
	d	a	f	d	a
	c	b	d	null	null

Tuple joint

4. Cas particuliers

L'**autojointure** est un cas de jointure d'une relation avec elle-même. Cela se passe comme si on avait 2 copies différentes d'une même relation. Les noms des attributs sont préfixés du nom de relation afin d'éviter toute ambiguïté.

La **semijointure** est un cas de jointure ne conservant que les attributs d'une des relations .

V. Les opérations de calculs et d'agrégats

On peut également ajouter des opérations de calcul et de regroupement en appliquant une fonction statistique sur les attributs des relations. De nombreuses requêtes ont en effet besoin de ces opérations.

A. Colonnes calculées

Certaines colonnes peuvent provenir d'un résultat de calcul. La mise en place de ces calculs peut être effectuée dans le cadre des projections.

On définit ici une nouvelle relation R pour laquelle les attributs B et C sont numériques (condition pour effectuer des calculs).

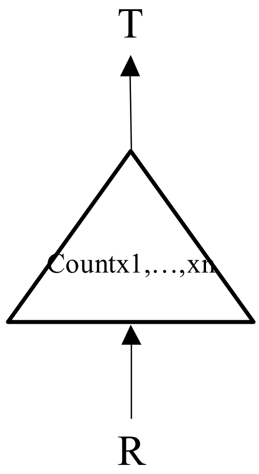
R	**A**	**B**	**C**
	a	2	10
	d	3	15
	c	5	5

	A	**B**	**BparC**	**MoitieC**
	a	2	2*10	10/2
	d	3	3*15	15/2
	c	5	5*5	5/2

T = PROJECTION (R , A, B, BparC=B*C, MoitieC=C/2)

T	**A**	**B**	**BparC**	**MoitieC**
	a	2	20	5
	d	3	45	7,5
	c	5	25	2,5

B. COMPTE



Compte est une opération courante qui permet de **DENOMBRER LES LIGNES** d'une relation qui ont une même valeur d'attribut en commun.

La relation résultante ne contient que les attributs de regroupement x_i choisis avec leurs occurrences dans la relation

Notation :

$$T = \text{compte } x_1, \ldots, x_n \text{ (R)}$$

ou

$$T = \text{COMPTE } (R , x_1,\ldots,x_n)$$

Si aucun attribut de regroupement n'est précisé, l'opération renvoie alors le nombre de tupes de R. La relation T ne comporte alors qu'un seul tuple avec une seule colonne.

R	A	B	C
	a	b	c
	d	a	f
	c	b	d

T	CompteR
	3

Exemple 1 : T = COMPTE (R)

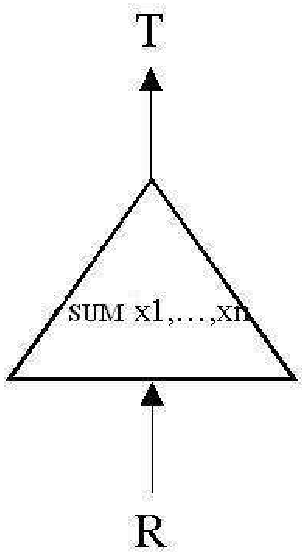
On compte le nombre de tuples de R

T	B	CompteR
	b	2
	a	1

Exemple 2 : T = COMPTE (R , B)

On compte le nombre de tuples de R par valeur différente de la colonne B

C. SOMME



Somme est une opération qui permet de faire la **SOMME CUMULEE** des valeurs d'un attribut Y pour chacune des valeurs différentes de attributs de regroupement x_1, …x_n.

Y doit être numérique.

La relation résultante ne contient que les différentes valeurs des attributs Xi de regroupement choisis ainsi que la somme cumulée des Y correspondants.

Notation :

T = somme x_1,…,x_n (R, Y)

ou

T = SOMME (R, Y , x_1,…,x_n)

Si aucun attribut de regroupement n'est précisé, l'opération renvoie alors la somme de toutes des valeurs de la colonne Y. La relation T ne comporte alors qu'un seul tuple avec une seule colonne.

Cette opération se généralise facilement à d'autres opérations : MOYENNE, MINIMUM, MAXIMUM.

On définit ici une nouvelle relation R pour laquelle l'attribut C est numérique.

R	A	B	C
	a	b	10
	d	a	15
	c	b	5

T	SommeC
	30

Exemple 1 : T = SOMME (R , C)

On effectue la somme de la colonne C pour tous les tuples de R

T	B	SommeC
	b	15
	a	15

Exemple 2 : T = SOMME (R , C, B)

On effectue la somme de la colonne C par valeur différente de la colonne B

VI. Les expressions de l'algèbre relationnelle

Une fois les opérateurs relationnels identifiés, il est alors facile de combiner ces opérations élémentaires pour construire des expressions de l'algèbre relationnelle permettant de fournir les réponses à des questions complexes.

Afin de s'approcher du langage SQL d'accès aux bases de données relationnelles, certains auteurs introduisent de nouveaux opérateurs.

Ainsi, vous trouverez un opérateur de classement des lignes (tri) :

- T = TRI (R, x_1 [croissant | decroissant], …,x_n[croissant | decroissant]),

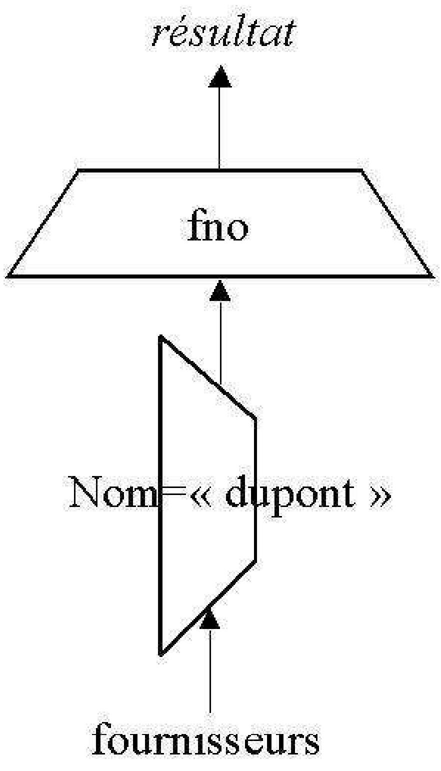
  qui produit une nouvelle relation T à partir de la relation R, en triant, en réordonnant, les lignes selon les critères de tri. Un critère de tri fait référence à une colonne et un mot clef qui précise le type de classement qu'on va appliquer à cette colonne (croissant, par défaut, ou décroissant).

A. Quelques exemples

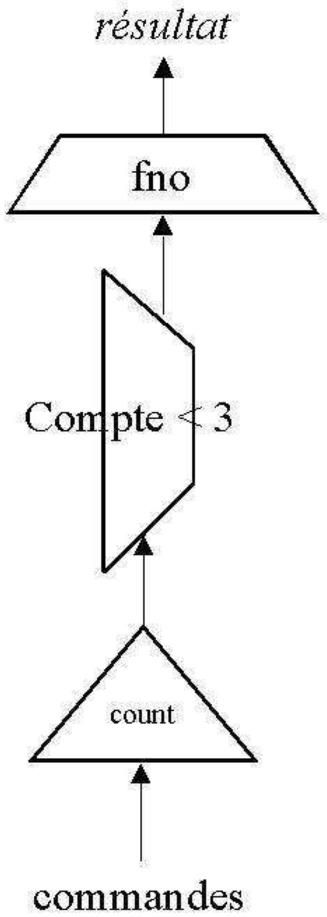
Exemple pour 3 relations :

- Fournisseurs (<u>fno</u>, nom, adresse, ville)
- Produits (<u>pno</u>, design, prix, poids, couleur)
- Commandes (<u>cno</u>, #fno, #pno, qute)

Déterminer les différents numéros de fournisseurs nommés "Dupont"

On sélectionne les données des fournisseurs dont le nom est 'dupont' : R1 = SELECTION (Fournisseurs , nom='dupont') On récupère seulement le numéro de fournisseur (colonne fno) : R2 = PROJECTION (R1 , fno) *Ou bien :* PROJECTION (SELECTION (Fournisseurs / nom= 'dupont') , fno)	

Déterminer les numéros de fournisseurs qui ont moins de 3 commandes

On compte le nombre de commandes par fournisseur : R1 = COMPTE (Commandes , fno) On sélectionne les lignes pour lesquelles le compte est inférieur à 3 : R2 = SELECTION (R1 , compteCommandes < 3) On récupère seulement le numéro de fournisseur (colonne fno): R3 = PROJECTION (R2 , fno) *Ou bien :* PROJECTION (SELECTION (COMPTE (Commandes , fno) , compte < 3) , fno)	

B. Plusieurs solutions

Il existe en général plusieurs requêtes algébriques différentes qui fournissent une solution à une question donnée. Si la relation obtenue est identique avec ces différentes requêtes, l'efficacité est rarement la même (*par exemple, l'opération de jointure est en général coûteuse ; la table temporaire la plus volumineuse pour effectuer cette requête est engendrée par le produit cartésien des 2 tables*).

Des calculs utilisant la cardinalité d'une relation ainsi que le nombre et la taille (moyenne) de ses attributs permet de choisir les requêtes la moins couteuse.

VII. Exercices d'entraînement

En utilisant les 3 relations (Fournisseurs, Produits, Commandes) :

- Fournisseurs (<u>fno</u>, nom, adresse, ville)
- Produits (<u>pno</u>, design, prix, poids, couleur)
- Commandes (<u>cno</u>, #fno, #pno, qte)

- lister les numéros et noms des fournisseurs
- lister les données sur les produits dont le poids est supérieur à 15kg
- liste les produits dont le poids est compris entre 15 et 40 kg
- lister les fournisseurs de Lille, Lyon ou Nice
- lister les noms des fournisseurs avec les numéros de produits commandés ainsi que la quantité commandée
- lister les couples de références de fournisseurs situés dans la même ville
- lister tous les produits de moins de 20 kg avec les quantités en cours de commande
- compter le nombre de commandes du produit no 102
- lister le nombre de commandes par fournisseur
- lister les numéros de fournisseur qui ont plus de 3 commandes d'au moins dix articles en cours

Algèbre relationnelle

Propositions de réponses :

- lister les numéros et noms des fournisseurs
  - **PROJECTION (Fournisseurs , fno, nom)**
- lister les données sur les produits dont le poids est supérieur à 15kg
  - **SELECTION (Produits , poids > 15)**
- liste les produits dont le poids est compris entre 15 et 40 kg
  - **SELECTION (Produits , poids >= 15 ET poids <=40)**
  - *Ou bien :*
  - **R1 = SELECTION (Produits , poids >= 15)**
  - **R2 = SELECTION (Produits , poids <= 40)**
  - **R3 = INTERSECTION (R1, R2)**
- lister les fournisseurs de Lille, Lyon ou Nice
  - **SELECTION (Fournisseurs , ville='Lille' OU ville='Lyon' OU ville='Nice')**
  - *Ou bien :*
  - **R1 = SELECTION (Fournisseurs , ville='Lille')**
  - **R2 = SELECTION (Fournisseurs ,ville='Lyon')**
  - **R3 = SELECTION (Fournisseurs , ville='Nice')**
  - **R4 = UNION (R1, R2)**
  - **R5 = UNION (R4, R3)**
- lister les noms des fournisseurs avec les numéros de produits commandés ainsi que la quantité commandée
  - **R1 = SOMME (Commandes , qte, fno, pno)**
  - **R2 = JOINTURE (Fournisseurs, R1 , Fournisseurs.fno = R1.fno)**
  - **R2 = PROJECTION (R1 , nom, pno, Sommeqte)**
- lister les couples de références de fournisseurs situés dans la même ville
  - **JOINTURE (Fournisseurs A, Fournisseurs B , A.ville = B.ville)**
- lister tous les produits de moins de 20 kg avec les quantités en cours de commande
  - **R1 = SELECTION (Produits , poids < 20)**
  - **R2 = SOMME (Commandes , qte , pno)**
  - **R3 = JOINTURE (R1, R2 , R1.pno = R2.pno)**
- compter le nombre de commandes du produit no 102
  - **R1 = SELECTION (Commandes , pno = 102)**
  - **R2 = COMPTE (R1)**
- lister le nombre de commandes par fournisseur
  - **R1 = COMPTE (Commandes , fno)**
  - **R2 = JOINTURE (Fournisseur, R1 , Fournisseur.fno = R1.fno)**
  - **R3 = PROJECTION (R2 , fno, nom, R2.compte)**
- lister les numéros de fournisseur qui ont plus de 3 commandes d'au moins dix articles en cours (on considère qu'un même numéro de commande peut être répété, c'est-à-dire, qu'il faudrait que l'identifiant comprenne un numéro de ligne)
  - **R1 = COMPTE (Commandes , fno, cno, pno)**
  - **R2 = SELECTION (R1 , R1.compte >= 10)**
  - **R3 = COMPTE (R2 , fno, cno)**
  - **R4 = SELECTION (R3 , R3.compte > 3)**
  - **R5 = PROJECTION (R4, fno)**

VIII. Bibliographie

Bases de données et systèmes d'information - le modèle relationnel : langages, systèmes et méthodes - Hacer Boudjlida - Dunod - 1999 - isbn 2 10 004309 9

des bases de données à l'internet - Philippe Mathieu - Vuibert Informatique - 2000 - isbn 2-7117-8669-2

SQL2 – de la théorie à l'application – Pierre Delmal – bibliothèque des universités – 1995 – isbn 2 8041 2179 8