Algèbre & Calculs relationnels

Bernard ESPINASSE
Professeur à Aix Marseille Université (AMU)
Ecole Polytechnique Universitaire de Marseille

Janvier 2018

Algèbre relationnelle

- **Les opérations de l'algèbre relationnelle**
- **Introduction au langage SQL**

Calculs relationnels

- **Calcul relationnel de tuples**
- **Introduction au langage QUEL**
- **Calcul relationnel de domaine**
- **Introduction au langage QBE**

Plan

Langages pour la manipulation d'une base de données relationnelle

Langages formels pour les bases de données :

- **algèbre relationnelle (Codd 70): langages algèbriques** définissant une collection d'opérations formelles sur les relations
- **calcul relationnel (Codd 72): langages à prédicats (prédicatifs)** définissant le résultat souhaité en utilisant des expressions de logique

Langages utilisateurs pour les bases de données :

- **inspirés de l'algèbre relationnelle :** *l'utilisateur spécifie une suite d'opérations à effectuer :*

  → ***langage SQL***

- **inspirés du calcul relationnel :** *l'utilisateur donne une définition du résultat cherché :*

  → ***langages QUEL et QBE***

1 - Algèbre relationnelle

- Les opérations de l'algèbre relationnelle
- Equivalences algébriques
- Arbres algébriques
- Exemples de requêtes algébriques
- Introduction au langage SQL

Algèbre

- **algèbre** = ensemble d'opérateurs de base, formellement définis, peuvant être combinés à souhait pour construire des expressions algébriques
- **algèbre fermée** : si le résultat de tout opérateur est du même type que les opérandes (ce qui est indispensable pour construire des expressions)
- **Complétude**: toute manipulation pouvant être souhaitée par les utilisateurs devrait pouvoir être exprimable par une expression algèbrique

Algèbre relationnelle

- **Opérandes** : relations **– tables** - du modèle relationnel (1NF)
- **Fermeture** : le résultat de toute opération est une nouvelle table
- **Complétude** : permet toute opération sauf les fermetures transitives
- **Opérations unaires** (1 opérande)
- **Opérations binaires** (2 opérandes)

Les opérations de l'algèbre relationnelle

Opérations de base :

- **Opérations ensemblistes :**
  - **union** ∪
  - **intersection** ∩
  - **différence** –
  - **produit cartésien** ×
- **Opérations spécifiques :**
  - **sélection/restriction** σ
  - **projection** π
  - **jointures** ⋈
  - **renommage** α

Opérations dérivées (obtenues par combinaison des opérations de base) :

- **division** ÷
- **jointure externe** ⟕
- **complément** ¬
- **fermeture transitive** τ

Sélection / Restriction

•Opération **unaire**, la **restriction** (ou sélection) d'une table **R** selon un critère de restriction ou qualification **Q** (pouvant porter sur un ou plusieurs attributs de R) est une table **R'** de même schéma que R dont les tuples sont des tuples de R vérifiant la qualification Q :

σ_Q (R) | RESTRICT (R, Q)

Exemple :

CLIENT	nocli	nom	ville
	121	BERTRAND	PARIS
	256	PAGNOL	MARSEILLE
	542	LANDRY	QUEBEC
	652	DUPOND	PARIS

Sélection des clients où ville = PARIS :

CLIENT-1 = $\sigma_{\text{ville = 'PARIS'}}$ (CLIENT) :

CLIENT-1	nocli	nom	ville
	121	BERTRAND	PARIS
	652	DUPOND	PARIS

Projection

- Opération **unaire** consistant à supprimer colonnes (attributs) de la table et en éliminant les tuples doubles (si un attribut de la clé primaire a été supprimé)
- la **projection** d'une table **R** de schéma (**a1, a2…ap,ap+1…an**) selon la direction (**a1, a2…ap**) est une table **S** de schéma (**a1, a2…ap**) dont les **tuples** sont ceux de R auxquels sont **supprimés les attributs n'appartenant pas à la direction de projection** et en **éliminant les tuples doubles :**

$\pi_{a1, a2...ap}$ R | PROJECT (R, a_1, a_2...a_p)

Exemple :

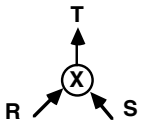
LIGNE_DE_COMMANDE	noCommande	noArticle	date	quantité
	10	121 A	5/5	10
	10	253 Z	5/5	1
	10	712 H	5/5	3
	12	253 Z	5/5	5
	13	712 H	6/5	2

LIGNE = $\pi_{\text{noArticle, date}}$ LIGNE_DE_COMMANDE :

LIGNE	noArticle	date
	121 A	5/5
	253 Z	5/5
	712 H	5/5
	712 H	6/5

Produit cartésien

Opération **binaire**, le **produit cartésien** de 2 tables **R** et **S** de schémas quelconques est une table **T** ayant pour attributs la **concaténation** de ceux de R et S et dont les **tuples** sont toutes les concaténations d'un tuple de R à un tuple de S :



T = R X S

PRODUCT (R, S)

Exemple :

VINS	n°	cru	millés.	deg.
	110	corbière	1974	13
	120	macon	1976	14

VITICULTEURS	nom	ville	région
	nicolas	poully	bougogne
	martin	bordeaux	bordelais

VINVIT = VINS X VITICULTEURS

VINVIT	n°	cru	millés.	deg.	nom	ville	région
	110	corbière	1974	13	nicolas	poully	bougogne
	120	macon	1976	14	martin	bordeaux	bordelais
	110	corbière	1974	13	martin	bordeaux	bordelais
	120	macon	1976	14	nicolas	poully	bougogne

Renommage α

Opération **unaire,** le **renommage** d'une table **R** permet d'obtenir une table **R'** dont les **tuples** sont ceux de R et dont le **schéma** est celui de R dans lequel **un attribut a été renommé** :

$$R' = \alpha\ [\text{nom\_attribut} : \text{nouveau\_nom}]\ R$$

- le schéma de R' = (α [n, m] R) est le même que le schéma (R) avec **n renommé en m**
- **précondition** : le nouveau nom d'attribut n'est pas déjà le nom d'un attribut de R
- **intérêt** : permet de résoudre des problèmes de compatibilité entre noms d'attributs de 2 tables opérandes d'une opération binaire :

Exemple :

R2 = α [B: C] R1

Jointure

Opération **binaire,** la **jointure** de 2 tables **R** et **S** est une table **T** obtenue ainsi :

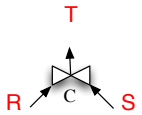
1. réaliser le **produit cartésien** des 2 tables R et S
2. effectuer une opération de **sélection** (ou qualification) entre un attribut de la table R et un attribut de la table S appelés "attributs de jointure"
3. effectuer ou non une opération de **projection** pour réduire le schéma de la table résultante

Remarques :

1. elle réalise une **concaténation de tables** limitée à des occurrences de tables présentant des valeurs communes sur des attributs de jointure
2. elle **matérialise le lien entre plusieurs tables** ou la fusion de plusieurs tables
3. la **sélection-qualification** ou "opérateur de jointure" est généralement **l'égalité**, mais peut être étendu à des opérateurs logiques quelconques.
4. elle **peut s'effectuer sur tout attribut**, sans préjuger de la pertinence sémantique du résultat obtenu, seules les jointures en égalité construites sur les attributs clés primaires traduisent des relations (conceptuelles).

Jointure

Opération **binaire,** la **jointure** de 2 table **R** et **S** selon une **condition** θ consiste à rapprocher les tuples de 2 tables **R** et **S** afin de former une troisième table **T** qui contient l'ensemble de tous le tuples obtenus en **concaténant** un tuple de R et un tuple de S vérifiant la condition θ :



R ⋈ S
Condition C

JOIN (R, S, C)

5. la condition de rapprochement θ est du type :

<attribut1> <**opérateur**> <attribut2>

avec attribut1 appartient à R et attribut2 appartient à S

6. on distingue selon l'opérateur :

- opérateur = : **équi-jointure**
- opérateur {<, >, ≤, ≥, ≠} : **inéqui-jointure** ou **théta-jointure**

Equi-jointure

Exemple d'équi-jointure :

VINS	n°	cru	millésime	deg.
	120	bordeaux	1975	14
	200	macon	1978	12
	210	macon	1977	12

VITICULTEUR	nom	ville	région
	nicolas	bordeaux	bordelais
	bernard	saumur	loire
	pierre	macon	bourgogne

VINVITI = VINS ⋈ VITICULTEUR

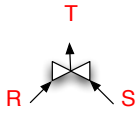
cru = ville

↓

VINVITI	n°	cru	millésime	deg.	nom	ville	région
	120	bordeaux	1975	14	nicolas	bordeaux	bordelais
	200	macon	1978	12	pierre	macon	bourgogne
	210	macon	1977	12	pierre	macon	bourgogne

Jointure naturelle

la **jointure naturelle** de 2 tables **R et S** est une table **T** dont les attributs sont l'union des attributs de R et de S et dont les **tuples** sont obtenus en concaténant un tuple de R et un tuple de S ayant **mêmes valeurs pour les attributs de même nom** :



R ⋈ S

JOIN (R, S)

Exemple :

CLIENT	noCli	nom
	121	DUVAL
	253	LEROY
	260	LANDRY
	293	SANCHEZ

COMMANDE	noCde	noCli	date
	10	121	5/5
	11	260	5/5
	12	121	5/5
	15	253	6/5

Jointure naturelle de CLIENT et COMMANDE (noCli de CLIENT = noCli de COMMANDE) :

CLICDE = CLIENT ⋈ COMMANDE = JOIN (CLIENT, COMMANDE)

CLICDE	noCli	nom	noCde	date
	121	DUVAL	10	5/5
	253	LEROY	15	6/5
	121	DUVAL	12	5/5
	260	LANDRY	11	5/5

Nonéqui-jointure ou Théta-jointure

L'inéqui-jointure ou **théta-jointure** de 2 tables **R et S selon** une **condition** de rapprochement **C, inégalité** portant entre un **attribut de R** et un **attribut de S** est une table **T** dont les **attributs** sont l'union des attributs de R et de S et dont les **tuples** sont obtenus en concaténant un tuple de R et un tuple de S **qui respectent** la condition C :
Exemple :

VINS	Cru	Millésime	Qualité
	Volnay	1983	A
	Volnay	1979	B
	Chablis	1983	A
	Julienas	1986	C

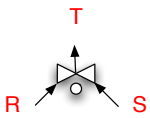
LIEU	Cru	Région	QualMoy
	Volnay	Bourgogne	A
	Chablis	Bourgogne	A
	Chablis	Californie	B

VINLIEU = VINS ⋈ [Qualité≠QualMoy] LIEU = JOIN (VINS, LIEU, Qualité≠QualMoy)

VINLIEU	Cru	Millésime	Qualité	Cru	Région	QualMoy
	Volnay	1983	A	Chablis	Californie	B
	Volnay	1979	B	Volnay	Bourgogne	A
	Volnay	1979	B	Chablis	Bourgogne	A
	Volnay	1983	A	Chablis	Californie	B
	Julienas	1986	C	Volnay	Bourgogne	A
	Julienas	1986	C	Chablis	Bourgogne	A
	Julienas	1986	C	Chablis	Californie	B

Jointure Externe

la **jointure externe** de 2 tables **R** et **S** est une table **T** obtenue par **jointure** de R et S et **ajout des tuples de R et de S ne participant pas à la jointure** avec des **valeurs nulles** pour les attributs de l'autre table :



R ⟕ S

EXT-JOIN (R, S)

Intérêt : composer des vues sans perte d'information

On distingue :

- **la jointure externe droite** (REXT-JOINT ⟖): elle garde seulement les tuples sans correspondant de la table de droite
- **la jointure externe gauche** (LEXT-JOINT ⟕): elle garde seulement les tuples sans correspondant de la table de gauche

Jointure Externe

Exemple :

VINS	Cru	Millésime	Qualité
	Volnay	1983	A
	Volnay	1979	B
	Julienas	1986	C

LIEU	Cru	Région	QualMoy
	Volnay	Bourgogne	A
	Chablis	Bourgogne	A
	Chablis	Californie	B

VINS-LIEU = VINS ⟕ LIEU = EXT-JOIN (VINS, LIEU) :

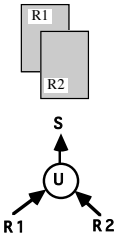
VINS-LIEU	Cru	Millésime	Qualité	Région	QualMoy
	Volnay	1983	A	Bourgogne	A
	Volnay	1979	B	Bourgogne	A
	Chablis	-	-	Bourgogne	A
	Chablis	-	-	Californie	B
	Julienas	1986	C	-	-

Opérations ensemblistes: Union, Intersection et Différence

Opérations binaires (tables = ensembles de tuples) :

- correspondant aux opérations habituelles de la **théorie des ensembles**
- ne peuvent être appliquées que sur des tables de **même schéma** et donnent une nouvelle table de même schéma :

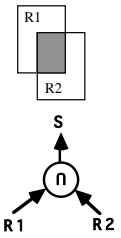
union



R1 ∪ R2

UNION (R1, R2)

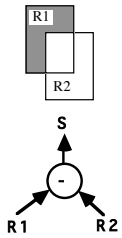
intersection



R1 ∩ R2

INTERSECT (R1, R2)

différence



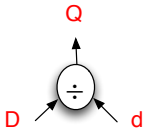
R1 - R2

MINUS (R1, R2)

Remarque : l'intersection est une opération dérivée car R1 ∩ R2 = R1 – (R1– R2)

Division

le quotient de la **division** d'une table **D (a1, a2…ap,ap+1…an**) par la sous table **d(ap … an)** est la **table Q(a1, a2…ap)** dont les **tuples** sont ceux qui concaténés à tout tuple de **d** donnent un tuple de **D :**



D ÷ d

DIVISION (D, d)

Intérêts de la division :

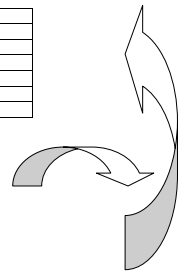
- elle permet de rechercher dans une table les sous-tables qui sont complétées par tous ceux d'une autre table
- elle permet ainsi de répondre à des requêtes de la forme « **quel que soit x, trouver y** »

Division

Exemple de division :

PIECE	nom	couleur	poids	ville
tuple 1	clou	rouge	14	Londres
tuple 2	boulon	vert	17	Paris
tuple 3	vis	bleu	12	Rome
tuple 4	vis	rouge	14	Londres
tuple 5	rivet	bleu	12	Paris
tuple 6	clou	bleu	12	Rome

NATURE	nom
tuple 1	vis
tuple 2	clou



Division de la table PIECE par la table NATURE :

PIECE_NATURE = PIECE ÷ NATURE :

PIECE_NATURE	couleur	poids	ville
tuple 1	bleu	12	Rome
tuple 2	rouge	14	Londres

Les tuples de la table PIECE_NATURE, concaténés à chacun des tuples de la table NATURE donnent un tuple de la table PIECE.

Equivalences algébriques

Quelques équivalences :

- **Commutativité de la sélection :** $\sigma_{e1}(\sigma_{e2}(T)) = \sigma_{e2}(\sigma_{e1}(T))$
- **Commutativités des intersections :** $R \cap S = S \cap R$
- **Commutativité de la jointure :** $R \bowtie S = S \bowtie R$
- **Eclatement d'une sélection conjonctive :** $\sigma_{e1 \text{ et } e2}(T) = \sigma_{e1}(\sigma_{e2}(T))$
- **Elimination des projections en cascades :** $\pi_{liste1}(\pi_{liste2}(T)) = \pi_{liste1}(T)$
- **Associativité de la jointure :** $R \bowtie (S \bowtie T) = (R \bowtie S) \bowtie T$
- **…**

D'où formulations équivalentes d'une même requête :

$\pi_{liste1}(\sigma_{e1}(R \bowtie S \bowtie T \bowtie V)) \Leftrightarrow \pi_{liste1}(\sigma_{e1}(((R \bowtie S) \bowtie T) \bowtie V))$

$\Leftrightarrow \pi_{liste1}(\sigma_{e1}(((R \bowtie T) \bowtie S) \bowtie V))$

Exemples de requêtes algébriques

Soit la base de données suivante :

JOURNAL (**code-j**, titre, prix, type, périodicité)
DEPOT (**no-dépôt**, nom-dépôt, adresse)
LIVRAISON (***no-dépôt, code-j,*** **date-liv**, quantité-livrée)

Requêtes algébriques :

- Quel est le prix des journaux ?

  $\pi_{[prix]}$ JOURNAL

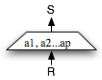
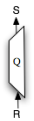
- Informations sur les journaux hebdomadaires ?

  $\sigma_{[périodicité = 'hebdomadaire']}$ JOURNAL

- Quels sont les codes des journaux livrés à Marseille ?

  $\pi_{[code\text{-}j]}(\sigma_{[adresse = 'Marseille']}\text{DEPOT} \bowtie \text{LIVRAISON})$

Conventions d'expression des opérateurs algébriques

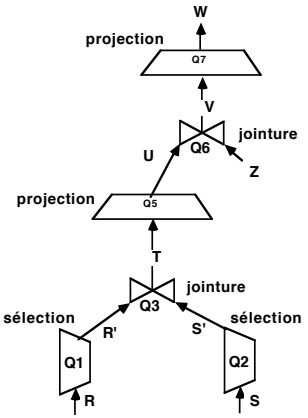
projection	$\pi_{a1, a2...ap}$ R		**union**	R ∪ S	
sélection-restriction	σ_{Q} (R)		**intersection**	R ∩ S	
jointure	R ⋈ S		**différence**	T = R – S	
produit cartésien	T = R X S		**division**	Q = D ÷ d	

- renomage de la colonne a2 en a2' : R' = α [a2 : a2'] R

Arbre algébrique

- exprimer une suite d'opération d'algèbre relationnelle associée à une requête:

Requête : $(\pi_{Q7}(\pi_{Q5}(\sigma_{Q1}(R) \bowtie_{[Q3]} \sigma_{Q2}(S)\ )) \bowtie_{[Q6]} Z)$



2 – Introduction au langage SQL

- Historique et structure de base d'une requête SQL
- Projection et sélection en SQL
- Produit cartésien et jointures en SQL
- Union, intersection et différence en SQL
- Connecteur logiques et fonctions de calcul en SQL

Historique et structure de base d'une requête SQL

Historique

- Acronyme de **Structured Query Langage**
- Dérivé de SEQUEL, interface de consultation pour System R (Astr IBM 81)
- Standard ANSI en 86, standard ISO en 87, amélioré en 1989 (**SQL 1**)
- Norme la plus courante (ANSI/ISO) en 92 (**SQL2**)
- Nouvelle norme 1999 (**SQL3**)

La structure de base d'une expression SQL comporte 3 clauses :

- **SELECT**: projection de l'algèbre relationnelle désigne la liste des attributs désirés
- **FROM**: désigne la liste des tables à considérer
- **WHERE**: désigne le prédicat à vérifier sur des attributs de table de la clause From

SELECT a1, a2,…, an **FROM** R1, R2 … Rm **WHERE** P

Soit :

$$\pi_{a1,\ a2,\ldots,\ an}\ \sigma_{P}\ (R1 \times R2 \times \ldots \times Rm)$$

BD relationnelle « fourniture »

PRODUIT (**npro**: no produit, **nomp**: nom produit, **qtes**: qté stock, **couleur** : couleur produit)

VENTE (**nvente**: no vente, **nomc**: nom client, ***nprv***: no produit vendu, **qtev**: qté vendue)

ACHAT (**nachat**: no achat, ***npra*** : no produit acheté, **qtea**: qté acheté, **nomf**: nom fournisseur)

PRODUIT	npro	nompro	qtes.	couleur
	1	chaise	60	rouge
	2	bureau	18	gris
	3	armoire	35	blanche
	4	lampe	51	jaune
	5	fauteuil	31	rouge
	6	chaise	20	verte
	7	lampe	3	rouge

VENTE	nvente	nomc	nprv	qtev
	10	Duval	1	10
	11	Landry	5	20
	12	Smith	4	10
	13	Duval	3	15
	15	Durant	1	20

ACHAT	nachat	date	npra	qtea	nomf
	10	4/9/02	5	10	Ribart
	11	5/9/02	4	20	Tellier
	12	8/9/02	3	10	Dupond
	13	12/9/02	2	15	Dupond
	15	16/9/02	1	20	Buvard

Projection et sélection en SQL

PRODUIT (**npro**, nomp, qtes, couleur)
VENTE (**nvente**, nomc, *nprv*, qtev)
ACHAT (**nachat**, *npra*, qtea, nomf)

- **projection**
  - `SELECT npro, couleur FROM PRODUIT` (avec doubles)
  - `SELECT UNIQUE npro, couleur FROM PRODUIT` (sans doubles)
- **Sélection**
  - `SELECT * FROM PRODUIT WHERE couleur = rouge AND qtes > 35`
  - `SELECT npro, qtes FROM PRODUIT WHERE couleur = rouge AND qtes > 35`
  - `SELECT npro, qtes FROM PRODUIT WHERE couleur = rouge AND qtes > 35 ORDER BY qtes ASC, couleur DESC`

Produit cartésien et jointures en SQL

- **Produit cartésien**

```
SELECT * FROM PRODUIT, VENTE
```

- **jointures**

```
SELECT * FROM PRODUIT, VENTE WHERE PRODUIT.npro =
 VENTE.nprv

SELECT nompro FROM PRODUITS WHERE npro IN

(SELECT nprv FROM VENTE)
```


Union, Intersection et Différence en SQL

- **union**

```
SELECT nprv FROM VENTE WHERE qtev > 15 UNION
SELECT nprv FROM VENTE WHERE nomc = Duval
```

sur une même table ou requêtes sur tables différentes avec même select !

- **intersection**

```
SELECT nprv FROM VENTE WHERE qtev > 15 INTERSECT
SELECT nopro FROM PRODUIT WHERE nompro = chaise
```

sur une même table ou requêtes sur tables différentes avec même select !

- **différence**

```
SELECT nopro FROM PRODUIT WHERE qtes > 50 MINUS
SELECT nopro FROM PRODUIT WHERE couleur = rouge
```

sur une même table ou requêtes sur tables différentes avec même select !

Connecteur logiques et fonctions de calcul en SQL

- **Connecteurs logiques dans SQL : AND, OR, NOT, IN, NOT IN**

```
SELECT nompro FROM PRODUIT WHERE npro NOT IN
   (SELECT nprv FROM VENTE WHERE qtev > 10
           AND nomc = Duval)
```

- **Fonctions de calculs : COUNT**: Nb de valeurs ; **SUM** : somme de valeurs ; **AVG** : moyenne de valeurs ; **MAX** : valeur maxi ; **MIN** : valeur mini

```
SELECT AVG (qtes) FROM PRODUIT WHERE couleur = rouge
```


Mise à jour de tuples en SQL

- **Insertion de tuples :**

PRODUIT (**npro**, nomp, qtes, couleur)

```
INSERT INTO PRODUIT VALUES ('P233', 'spatule', 30,
'orange') ;
```

- **Suppression et modification de tuples :**

```
DELETE PRODUIT WHERE couleur = rouge AND qtes < 5

UPDATE VENTE SET Qtev = 0 WHERE nomc = Bergman)
```

3 - Calculs relationnels

- Rappel sur le calcul des prédicats du 1° ordre
- Calcul relationnel de tuples
- Introduction au langage QUEL
- Calcul relationnel de domaine
- Introduction au langage QBE

Introduction au calcul Relationnel

- **Algèbre relationnelle :**
  - langage procédural permettant d'expliciter une séquence d'opérations qui conduiront à un résultat désiré.
- **Calcul relationnel :**
  - langage non-procédural permettant d'expliciter le résultat que l'on désire sans spécifier la séquence des opérations à effectuer.

2 types de calcul relationnel:

- **Calcul relationnel de Tuples :**

  variables = **tuples**

  → *langage **QUEL** (SGBD INGRES)*
- **Calcul relationnel de domaines :**

  variables = **valeurs** prises sur un **domaine**

  → *langage **QBE***

Rappel sur le calcul des prédicats du 1° ordre

- prédicats du 1°ordre = système formel de la logique
- les principaux langages de manipulation de données relationnels permettant le calcul relationnel sont basés sur cette logique

Syntaxe :

()	**parenthèses**
⊃	**implication**
¬	**non**
a,b	**constante**
x,y	**variables**
P,Q	**predicat à argument**
f,g,h	**fonction à argument**

Exemple de **formule bien formées** (NILSSON 71) :

¬P (a, g (a,b,a))
P (a,b) ⊃ (¬Q(c))
¬P (a) ⊃ Q (f(a))

en base de données on se sert en général de :

∀, ∃, et (∧), ou (∨), ¬

Calcul relationnel de Tuples

Langage d'interrogation de données formel permettant d'exprimer des questions à partir de formules dont les **variables** sont interprétées comme variant sur les **tuples** d'une table

Base de données exemple « fourniture »:

PRODUIT (**npro**, nomp, qtes, couleur)
VENTE (**nvente**, nomc, *nprv*, qtev)
ACHAT (**nachat**, *npra*, qtea, nomf)

Exemple de requêtes en calcul relationnel de tuples :

- {P.NOMP, P.COULEUR / PRODUIT (P)}

  → *liste des noms et des couleurs de tous les produits*
- {P.NOMP, P.QTES / PRODUIT (P) (P.COULEUR = "rouge")}

  → *nom et quantités en stock des produits de couleur rouge*
- {P.NOMP, A.NOMF, V.NOMC / PRODUIT (P) ∧ ACHAT (A) ∧ VENTE (V) (P.QTES > 100) ∧ (P.COULEUR = "rouge") ∧ (P.NPRO = V.NPRV) ∧ (P.NPRO = A.NPRA) }

  → *donner pour chaque produit en stock en quantité supérieure à 100 et de couleur rouge, les triplets nom de fournisseur ayant vendu ce produit, et nom de client ayant acheté ce produit.*

Introduction au langage QUEL (Ingres)

- **Langage QUEL (Zook 77), Ingres (Stonebraker 76)**
  - dérivé du calcul relationnel de tuples
  - pas de quantificateur (∀, ∃) explicites
  - + fonctions de calcul (moyenne, compteur, somme, min, max, ...)
  - + fonctions de mises à jour sophistiquées
  - vues, …
- **Déclaration des variables dans QUEL :**

  séparation de la définition des variables de l'expression de la requête :

  RANGE OF variable IS nom_de_relation

  RANGE OF p IS produit ; RANGE OF a, b, c IS r1, r2, r3
- **Structure de base d'une expression QUEL :**

  RANGE OF t IS r : où t = variable de tuple dont la valorisation est restreinte aux tuples de la table r

  RETRIEVE : (extraire) équivalent à la clause SELECT de SQL

  WHERE : prédicat suivant lequel s'effectue l'extraction

Langage QUEL (Ingres)

- **Recherche en QUEL :**

  RETRIEVE [[INTO] nom_table] (liste_résultat)[**WHERE** qualification]

  Ex: RANGE p IS produit, a IS chat, v IS vente)
  - RETRIEVE (p.nomp, p.couleur)
  - RETRIEVE (p.nomp, p.qtés) WHERE p.couleur="rouge"
  - RETRIEVE (p.nomp, a.nomf) WHERE p.couleur="rouge"
  - RETRIEVE (p.nomp, a.nomf, v.nomc) WHERE (p.qtes>100) ∧ (p.couleur="rouge") ∧ (p.npro=v.nprv) ∧ (p.npro=a.npra)
- **Tri : RETRIEVE ... WHERE ... SORT BY ...**
  - RANGE e IS employé, d IS département
  - RETRIEVE (e.nom, e.salaire, e.num_departement) WHERE e.salaire>300000 SORT BY nom
- **Mises à jour en QUEL : REPLACE, APPEND TO, DELETE**
  - REPLACE employé(salaire=180000) [WHERE employé.nom="Durant"]
  - APPEND TO département (dname="ventes", étage=3, batiment="A", division="D1")
  - DELETE d WHERE d.nom="ventes"

Calcul relationnel de domaine

Langage d'interrogation de données formel permettant d'exprimer des questions à partir de formules bien formées dont **chaque variable est interprétée** comme variant sur le **domaine** d'un **attribut** d'une table

Base de données exemple « fourniture »:

PRODUIT (**npro**, nomp, qtes, couleur)
VENTE (**nvente**, nomc, *nprv*, qtev)
ACHAT (**nachat**, *npra*, qtea, nomf)

Exemple de requêtes en calcul relationnel de domaines :

- {x, y / PRODUIT(nomp:x, couleur:y)}

  → *liste des noms et des couleurs de tous les produits*
- {x, y / PRODUIT(nomp: x, qtes: y, couleur: "rouge")}

  → *nom et quantités en stock des produits de couleur rouge*
- {x, y, z / PRODUIT(npro: t, nom: x, qtes: u, couleur: "rouge") ∧ ACHAT(npra: t, nomf: y) ∧ VENTE(nprv: t, nomc: z) ∧ (u>100)}

  → *donner pour chaque produit en stock en quantité supérieure à 100 et de couleur rouge, les triplets nom de fournisseur ayant vendu ce produit, et nom de client ayant acheté ce produit.*

Le langage QBE (1)

QBE : Query By Example (Zloff 77, IBM Yorktown Heights 78

- **mise en oeuvre visuelle** (tableaux affichés) du calcul relationnel de domaine : **QMF** (SQL/DS et DB2)
- **idée de base** : faire formuler une requête à l'utilisateur par un exemple d'une réponse possible à la question :

  1- affichage de schémas relationnels vides à l'écran

  2- expression des requêtes par remplissage des schémas

PRODUIT	npro	nomp	qtes	couleur
		P.	P.	rouge

  → *nom et quantités en stock des produits de couleur rouge*
- **Conventions :**
  - **P. : attributs à projeter**
  - **constantes tapées directement, précédées d'un comparateur ((= par défaut), >, <, ≥, ≤, ¬, ..)**
  - **variables : soulignées**
  - **∀: all**
  - **∧ (ou) : 2 lignes superposées (formules normale disjonctives)**

le langage QBE (2)

- **Requêtes multi-tables :**

→ noms des clients ayant acheté au moins un produit de couleur verte ?

PRODUIT	npro	nomp	qtes	couleur
	X			verte

VENTE	nvente	nomc	nprv	qtes
		P.	X	

- **avec non élimination des doublons : ALL**

VENTE	nvente	nomc	nprv	qtes
		P.ALL.X		

- **Tri (DO : descendant; AO: ascendant) :**

PRODUIT	npro	nomp	qtes	couleur
	P.DO.X		>100	

Le langage QBE (3)

- **fermeture transitive de tables :**

NOMENCLATURE	COMPOSE	COMPOSANT
	voiture	chassis
	voiture	moteur
	voiture	batterie
	moteur	piston
	moteur	bielle
	moteur	carter
	piston	boulon
	piston	écrou

→ recherche des composants de 2° niveau

NOMENCLATURE	COMPOSE	COMPOSANT
	voiture	x
	x	P. y

→ recherche des composants de 3 °niveau (3L = level 3):

NOMENCLATURE	COMPOSE	COMPOSANT
	voiture	P. y (3L)

Le langage QBE: Mises à jour

- **Insertions (Insert):**

PRODUIT	npro	nomp	qtes	couleur
I.	8	tableau	100	verte

→ insertion du tuple <8, tableau, 100, verte>

- **Mise à jour (Update) :**

PRODUIT	npro	nomp	qtes	couleur
U.	x		y	rouge
	x		y+100	

→ mise à jour des quantités en stock des produits rouges

- **Suppression (Delete) :**

PRODUIT	npro	nomp	qtes	couleur
D.				rouge

→ suppression des produits de couleur rouge