Introduction aux Bases de Données

Maîtrise de Sciences Cognitives
Année 2003-2004
Jérôme Darmont
http://eric.univ-lyon2.fr/~jdarmont/

Plan du cours

I. Introduction
II. Le modèle UML
III. Le modèle relationnel

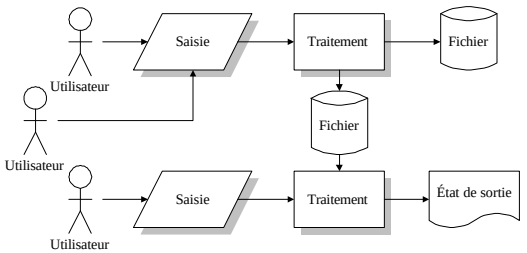
Plan du cours

☞ **I. Introduction**
II. Le modèle UML
III. Le modèle relationnel

I.1. Historique des bases de données

- Jusqu'aux années 60 : organisation classique en **fichiers**
- Fin des années 60 : apparition des premiers SGBD (*Systèmes de Gestion de Bases de Données*), les **systèmes réseaux** et **hiérarchiques**
- À partir de 1970 : deuxième génération de SGBD, les **systèmes relationnels**
- Début des années 80 : troisième génération de SGBD, les **systèmes orientés objet**

I.2. Limites des systèmes à fichiers

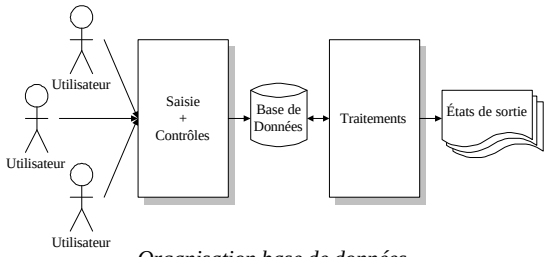


Organisation en fichiers

I.2. Limites des systèmes à fichiers

- Particularisation de la saisie et des traitements en fonction des fichiers ⇒ un ou plusieurs programmes par fichier
- Contrôle en différé des données ⇒ augmentation des délais et du risque d'erreur
- Particularisation des fichiers en fonction des traitements ⇒ grande redondance des données

I.3. Organisation base de données



Organisation base de données

I.3. Organisation base de données

- Uniformisation de la saisie et standardisation des traitements (ex. tous les résultats de consultation sous forme de listes et de tableaux)
- Contrôle immédiat de la validité des données
- Partage de données entre plusieurs traitements
  $\Rightarrow$ limitation de la redondance des données

I.4. Définitions

- **Base de données (BD) :** Collection de données cohérentes et structurées
- **Système de Gestion de Bases de Données (SGBD) :** Logiciel(s) assurant structuration, stockage, maintenance, mise à jour et consultation des données d'une BD

I.5. Objectifs des SGBD

- **Indépendance physique :** un remaniement de l'organisation physique des données n'entraîne pas de modification dans les programmes d'application (traitements)
- **Indépendance logique :** un remaniement de l'organisation logique des fichiers (ex. nouvelle rubrique) n'entraîne pas de modification dans les programmes d'application non concernés

I.5. Objectifs des SGBD

- **Manipulation facile des données :** un utilisateur non-informaticien doit pouvoir manipuler simplement les données (interrogation et mise à jour)
- **Administration facile des données :** un SGBD doit fournir des outils pour décrire les données, permettre le suivi de ces structures et autoriser leur évolution (tâche de l'*administrateur BD*)

I.5. Objectifs des SGBD

- **Efficacité des accès aux données :** garantie d'un bon *débit* (nombre de transactions exécutées par seconde) et d'un bon *temps de réponse* (temps d'attente moyen pour une transaction)
- **Redondance contrôlée des données :** diminution du volume de stockage, pas de mise à jour multiple ni d'incohérence

I.5. Objectifs des SGBD

- **Cohérence des données :** ex. L'âge d'une personne doit être un nombre entier positif. Le SGBD doit veiller à ce que les applications respectent cette règle (*contrainte d'intégrité*).
- **Partage des données :** utilisation simultanée des données par différentes applications
- **Sécurité des données :** les données doivent être protégées contre les accès non-autorisés ou en cas de panne

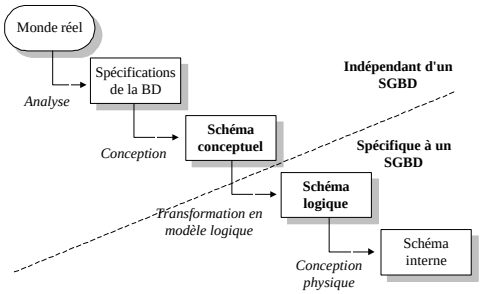
I.6. Fonctions des SGBD

- Description des données : *Langage de Définition de Données* (LDD)
- Recherche des données
- Mise à jour des données
- Transformation des données

} *Langage de Manipulation de Données* (LMD)

- Contrôle de l'intégrité des données
- Gestion de transactions et sécurité

I.7. Processus de conception d'une base de données



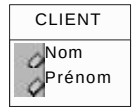
Plan du cours

II.1. Généralités

- UML = *Unified Modeling Language*
- Ensemble de *formalismes graphiques* pour la modélisation *orientée objet* (analyse)
- Auteurs : J. Rumbaugh, G. Booch, I. Jacobson
- Standard de l'OMG (*Object Management Group*) depuis 1997, standard de fait soutenu par *Rational Software*, Microsoft, Hewlett-Packard, Oracle, IBM…
- Mise en œuvre d'une BD : transformation d'un *diagramme de classes* UML en schéma logique

II.2. Classes et attributs

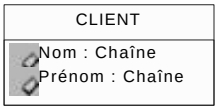
- **Classe :** Groupe d'entités du monde réel ayant les mêmes caractéristiques et le même comportement (ex. CLIENT)



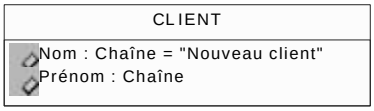
- **Attribut :** Propriété de la classe (ex. Nom et Prénom du client)

II.2. Classes et attributs

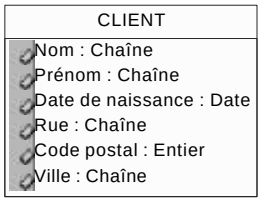
- **Type d'attribut :**
  - Entier
  - Réel
  - Chaîne de caractères
  - Date



- **Valeur par défaut** (=)



II.2. Classes et attributs



Exemple de classe avec ses attributs

II.3. Instances

- **Classe :** ex. CLIENT
- **Instances** (objets) de la classe CLIENT : les clients

  Albert Dupont
  James West
  Marie Martin
  Gaston Durand
  ...

II.4. Identifiants

Liste des clients

Nom	*Prénom*	*Date de Naissance*	*Etc.*
Dupont	Albert	01/06/70	...
West	James	03/09/63	...
Martin	Marie	05/06/78	...
Durand	Gaston	15/11/80	...
Titgoutte	Justine	28/02/75	...
Dupont	Noémie	18/09/57	...
Dupont	Albert	23/05/33	...

Problème : Comment distinguer les Dupont ?

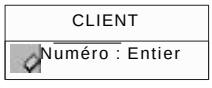
II.4. Identifiants

Solution : Ajouter un attribut *Numéro de client*

Numéro	*Nom*	*Prénom*	*Date Naissance*
1	Dupont	Albert	01/06/70
2	West	James	03/09/63
3	Martin	Marie	05/06/78
4	Durand	Gaston	05/11/80
5	Titgoutte	Justine	28/02/75
6	Dupont	Noémie	18/09/57
7	Dupont	Albert	23/05/33

II.4. Identifiants

- Le numéro de client est un **identifiant**. Un identifiant caractérise *de façon unique* les instances d'une classe.
- NB : Dans le paradigme objet, un objet est *déjà identifié* par son OID (*Object IDentifier*). La finalité de notre utilisation d'UML n'étant pas OO, nous ajouterons un attribut identifiant.
- **Convention graphique :**
  NB : Ne pas confondre avec les attributs de classe UML dont c'est la notation usuelle



II.5. Associations et multiplicité

Association : liaison perçue entre des classes

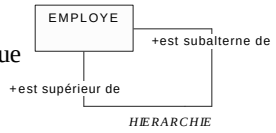
ex. Les clients commandent des produits.

- Les classes CLIENT et PRODUIT peuvent être qualifiées de *participantes* à l'association COMMANDE.
- *Degré* ou *arité* d'une association = nombre de classes participantes. En général : *associations binaires* (de degré 2).

II.5. Associations et multiplicité

Associations récursives et rôles

- *Association récursive :* une même instance de classe peut jouer plusieurs rôles dans la même association (ex. employés et supérieurs hiérarchiques).
- *Rôle :* fonction de chaque classe participante (+).



II.5. Associations et multiplicité

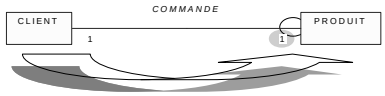
Multiplicité (ou cardinalité)

- **Définition :** Indicateur qui montre combien d'instances de la classe considérée peuvent être liées à une instance de l'autre classe participant à l'association
  - 1 Un et un seul
  - 0..1 Zéro ou un
  - 0..* ou * Zéro ou plus
  - 1..* Un ou plus
  - M..N De M à N (M, N entiers)

II.5. Associations et multiplicité

Association « 1-1 »

ex. Un client donné ne commande qu'un seul produit. Un produit donné n'est commandé que par un seul client.



Lire : Un client commande *multiplicité* (1) produit(s).

II.5. Associations et multiplicité

Association « 1-N »

ex. Un client donné commande plusieurs produits. Un produit donné n'est commandé que par un seul client.

La multiplicité « un à plusieurs » (1..*) peut aussi être « zéro à plusieurs » (0..* ou *).

II.5. Associations et multiplicité

Association « 0 ou 1-N »

ex. Un client donné commande plusieurs produits. Un produit donné est commandé au maximum par un client, mais peut ne pas être commandé.

La multiplicité « un à plusieurs » (1..*) peut aussi être « zéro à plusieurs » (0..* ou *).

II.5. Associations et multiplicité

Association « M-N »

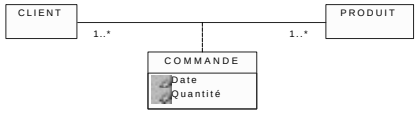
ex. Un client donné commande plusieurs produits. Un produit donné est commandé par plusieurs clients.

Les multiplicités « un à plusieurs » (1..*) peuvent aussi être « zéro à plusieurs » (0..* ou *).

II.5. Associations et multiplicité

Classe-association : Dans une *association M-N*, il est possible de caractériser l'association par des attributs (qui doivent appartenir à une classe).

ex. Une commande est passée à une Date donnée et concerne une Quantité de produit fixée.



II.6. Exemple VPC complet

Spécifications

- Les clients sont caractérisés par un numéro de client, leur nom, prénom, date de naissance, rue, code postal et ville.
- Ils commandent des produits à une date donnée et dans une quantité donnée.

II.6. Exemple VPC complet

Spécifications (suite)

- Les produits sont caractérisés par un numéro de produit, leur désignation et leur prix unitaire.
- Chaque produit est fourni par un fournisseur unique (mais un fournisseur peut fournir plusieurs produits).
- Les fournisseurs sont caractérisés par un numéro de fournisseur et leur raison sociale.

II.6. Exemple VPC complet

Marche à suivre pour produire un diagramme de classes UML :

1) Identifier les classes.
2) Identifier les associations entre les classes.
3) Identifier les attributs de chaque classe et de chaque classe d'association.
4) Évaluer la multiplicité des associations.

II.6. Exemple VPC complet

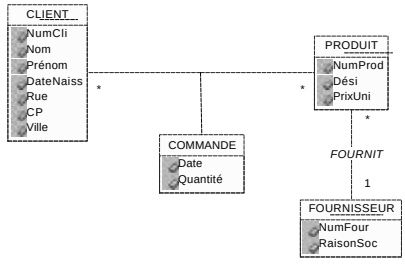


Diagramme de classes UML de l'exemple VPC

Plan du cours

I. Introduction

II. Le modèle E/A

☞ **III. Le modèle relationnel**

III.1. Généralités

- Le modèle relationnel est un *modèle logique* associé aux SGBD relationnels (ex. Oracle, DB2, SQLServer, **Access**, Paradox, dBase…).
- **Objectifs du modèle relationnel :**
  - indépendance physique
  - traitement du problème de redondance des données
  - LMD non procéduraux (faciles à utiliser)
  - devenir un standard

III.2. Relations, attributs, n-uplets

- Une *relation* R est un ensemble d'*attributs* $\{A_1, A_2, \ldots, A_n\}$.
  ex. La relation PRODUIT est l'ensemble des attributs {NumProd, Dési, PrixUni}
- Chaque attribut A_i prend ses valeurs dans un *domaine* $dom(A_i)$.
  ex. PrixUni $\in$]0, 10.000]

III.2. Relations, attributs, n-uplets

- Un *n-uplet* t est un ensemble de valeurs $t = <V_1, V_2, \ldots, V_n>$ où $V_i \in dom(A_i)$ ou V_i est la valeur nulle (NULL).
  ex. <112, 'Raquette de tennis', 300> est un n-uplet de la relation PRODUIT.
- **Notation :** R $(A_1, A_2, \ldots, A_n)$
  ex. PRODUIT (NumProd, Dési, PrixUni)

III.3. Contraintes d'intégrité

- **Clé primaire :** Ensemble d'attributs dont les valeurs permettent de distinguer les n-uplets les uns des autres (*identifiant*).
  ex. NumProd clé primaire de la relation PRODUIT
- **Clé étrangère :** Attribut qui est clé primaire d'une autre relation.
  ex. Connaître le fournisseur de chaque produit ⇒ ajout de l'attribut NumFour à la relation PRODUIT

III.3. Contraintes d'intégrité

Notations : clés primaires soulignées, clés étrangères *en italiques*
ex. PRODUIT (NumProd, Dési, PrixUni, *NumFour*)

- **Contraintes de domaine :** les attributs doivent respecter une condition logique
  ex. PrixUni > 0 ET PrixUni ≤ 10000

III.4. Traduction UML-Relationnel

1) Chaque classe devient une relation. Les attributs de la classe deviennent attributs de la relation. L'identifiant de la classe devient clé primaire de la relation.

ex. CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)

III.4. Traduction UML-Relationnel

2) Chaque association 1-1 est prise en compte en incluant la clé primaire d'une des relations comme clé étrangère dans l'autre relation.

ex. Si un client peut posséder un compte, on aura :

COMPTE (NumCom, Solde)
CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville, *NumCom*)

III.4. Traduction UML-Relationnel

3) Chaque association 1-N est prise en compte en incluant la clé primaire de la relation dont la multiplicité maximale est * comme clé étrangère dans l'autre relation.

ex.

PRODUIT (NumProd, Dési, PrixUni, *NumFour*)
FOURNISSEUR (NumFour, RaisonSoc)

III.4. Traduction UML-Relationnel

4) Chaque association M-N est prise en compte en créant une nouvelle relation dont la clé primaire et la concaténation des clés primaires des relations participantes. Les attributs de la classe d'association sont insérés dans cette nouvelle relation si nécessaire.

ex. COMMANDE (*NumCli*, *NumProd*, Date, Quantité)

III.4. Traduction UML-Relationnel

Schéma relationnel complet de l'exemple VPC

CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville)
PRODUIT (NumProd, Dési, PrixUni, *NumFour*)
FOURNISSEUR (NumFour, RaisonSoc)

COMMANDE (*NumCli*, *NumProd*, Date, Quantité)

III.5. Problème de la redondance

[En dehors des clés étrangères]

ex. Soit la relation COMMANDE_PRODUIT.

NumProd	*Quantité*	*NumFour*	*Adresse*
101	300	901	Quai des brumes
104	1000	902	Quai Cl. Bernard
112	78	904	Quai des Marans
103	250	901	Quai des brumes

Cette relation présente différentes anomalies.

III.5. Problème de la redondance

- **Anomalies de modification :** Si l'on souhaite mettre à jour l'adresse d'un fournisseur, il faut le faire pour tous les n-uplets concernés.
- **Anomalies d'insertion :** Pour ajouter un fournisseur nouveau, il faut obligatoirement fournir des valeurs pour *NumProd* et *Quantité*.
- **Anomalies de suppression :** ex. La suppression du produit 104 fait perdre toutes les informations concernant le fournisseur 902.

III.6. Normalisation

Objectifs de la normalisation :

- Suppression des problèmes de mise à jour
- Minimisation de l'espace de stockage (élimination des redondances)

III.6. Normalisation

Les dépendances fonctionnelles (DF)

Soit R (X, Y, Z) une relation où X, Y, et Z sont des ensembles d'attributs. Z peut être vide.

Définition : *Y dépend fonctionnellement de X* ($X \rightarrow Y$) si c'est toujours la même valeur de Y qui est associée à X dans la relation R.

ex. PRODUIT (NumProd, Dési, PrixUni)
NumProd $\rightarrow$ Dési, Dési $\rightarrow$ PrixUni

III.6. Normalisation

Propriétés des dépendances fonctionnelles

- *Réflexivité :* Si $Y \subseteq X$ alors $X \rightarrow Y$.
- *Augmentation :* Si $W \subseteq Z$ et $X \rightarrow Y$ alors $X, Z \rightarrow Y, W$.
- *Transitivité :* Si $X \rightarrow Y$ et $Y \rightarrow Z$ alors $X \rightarrow Z$.

(Règles d'inférence d'Armstrong)

III.6. Normalisation

Propriétés des dépendances fonctionnelles

- *Pseudo-transitivité :* Si $X \rightarrow Y$ et $Y, Z \rightarrow T$ alors $X, Z \rightarrow T$.
- *Union :* Si $X \rightarrow Y$ et $X \rightarrow Z$ alors $X \rightarrow Y, Z$.
- *Décomposition :* Si $Z \subseteq Y$ et $X \rightarrow Y$ alors $X \rightarrow Z$.

NB : La notation X, Y signifie $X \cup Y$.

III.6. Normalisation

Première forme normale (1FN)

Une relation est en 1FN si tout attribut n'est pas décomposable.

ex. Les relations PERSONNE (Nom, Prénoms, Age) et DEPARTEMENT (Nom, Adresse, Tel) ne sont pas en 1FN si les attributs *Prénoms* et *Adresse* peuvent être du type [Jean, Paul] ou [Rue de Marseille, Lyon].

III.6. Normalisation

Deuxième forme normale (2FN)

Une relation est en 2FN si :
- elle est en 1FN ;
- tout attribut non clé primaire est dépendant de la clé primaire entière.

ex. La relation CLIENT (NumCli, Nom, Prénom, DateNaiss, Rue, CP, Ville) est en 2FN.

III.6. Normalisation

Deuxième forme normale (2FN)

ex. La relation COMMANDE_PRODUIT (NumProd, Quantité, NumFour, Ville) n'est pas en 2FN car on a NumProd, NumFour → Quantité et NumFour → Ville.

La décomposition suivante donne deux relations en 2FN : COMMANDE (NumProd, NumFour, Quantité) ; FOURNISSEUR (NumFour, Ville).

III.6. Normalisation

Troisième forme normale (3FN)

Une relation est en 3FN si :
- elle est en 2FN ;
- il n'existe aucune DF entre deux attributs non clé primaire.

ex. La relation CINEMA (NoFilm, NoRéal, Nom) avec les DF NoFilm → NoRéal, NoRéal → Nom et NoFilm → Nom est en 2FN, mais pas en 3FN.

III.6. Normalisation

Troisième forme normale (3FN)

Anomalies de mise à jour sur la relation CINEMA : il n'est pas possible d'introduire un nouveau réalisateur sans préciser le film correspondant.

La décomposition suivante donne deux relations en 3FN qui permettent de retrouver (par transitivité) toutes les DF : R1 (NoFilm, NoRéal)
R2 (NoRéal, Nom).

III.7. Algèbre relationnelle

- Ensemble d'opérateurs qui s'appliquent aux relations
- Résultat : nouvelle relation qui peut à son tour être manipulée

⇒ L'algèbre relationnelle permet d'effectuer des recherches dans les relations.

III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Union :* T = R ∪ S ou T = UNION (R, S)
  R et S doivent avoir même schéma.

  ex. R et S sont les relations PRODUIT de deux sociétés qui fusionnent et veulent unifier leur catalogue.

  Notation graphique :

III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Intersection :* T = R ∩ S ou
  T = INTERSECT (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de trouver les produits communs aux catalogues de deux sociétés.

  Notation graphique : 

III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Différence :* T = R - S ou T = MINUS (R, S)
  R et S doivent avoir même schéma.

  ex. Permet de retirer les produits de la relation S existant dans la relation R.

  Notation graphique : 

III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Produit cartésien :* T = R x S
  ou T = PRODUCT (R, S)

  Associe chaque n-uplet de R à chaque n-uplet de S.

  Notation graphique : 

III.7. Algèbre relationnelle

ex.

NumProd	*Dési*
0	P1
1	P2

X

NumFour	*RaisonSoc*
10	F1
20	F2
30	F3

=

NumProd	*Dési*	*NumFour*	*RaisonSoc*
0	P1	10	F1
1	P2	10	F1
0	P1	20	F2
1	P2	20	F2
0	P1	30	F3
1	P2	30	F3

III.7. Algèbre relationnelle

Opérateurs ensemblistes

- *Division :* T = R ÷ S ou T = DIVISION (R, S)

  R (A_1, A_2, …, A_n) S (A_{p+1}, …, A_n)
  T (A_1, A_2, …, A_p) contient tous les n-uplets tels que leur concaténation à *chacun* des n-uplets de S donne toujours un n-uplet de R.

  Notation graphique : 

III.7. Algèbre relationnelle

ex.

NumCli	*Date*	*Quantité*
1	22/09/99	1
1	22/09/99	5
1	10/10/99	2
2	15/10/99	9
3	22/09/99	1
3	10/10/99	2

÷

Date	*Quantité*
22/09/99	1
10/10/99	2

=

NumCli
1
3

III.7. Algèbre relationnelle

Opérateurs spécifiques

- *Projection :* T = Π <A, B, C> (R)
  ou T = PROJECT (R / A, B, C)

  T ne contient que les attributs A, B et C de R.
  ex. Noms et prénoms des clients.
  Notation graphique :

III.7. Algèbre relationnelle

Opérateurs spécifiques

- *Restriction :* T = σ <C> (R)
  ou T = RESTRICT (R / C)

  T ne contient que les attributs de R qui satisfont la condition C.
  ex. C = Clients qui habitent Lyon.
  Notation graphique :

III.7. Algèbre relationnelle

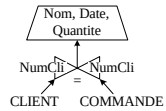
Opérateurs spécifiques

- *Jointure naturelle :* T = R >< S
  ou T = JOIN (R, S)

  Produit cartésien R x S et restriction A = B sur les attributs A ∈ R et B ∈ S.
  Notation graphique :

III.7. Algèbre relationnelle

ex. Commandes avec le nom du client et pas seulement son numéro



NB : *Requête* = enchaînement d'opérations

III.7. Algèbre relationnelle

Décomposition des opérations

CL

NumCli	*Nom*
1	C1
2	C2
3	C3

X

CO

NumCli	*Date*	*Quantité*
1	22/09/99	1
3	22/09/99	5
3	22/09/99	2

III.7. Algèbre relationnelle

=

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
2	C2	1	22/09/99	1
3	C3	1	22/09/99	1
1	C1	3	22/09/99	5
2	C2	3	22/09/99	5
3	C3	3	22/09/99	5
1	C1	3	22/09/99	2
2	C2	3	22/09/99	2
3	C3	3	22/09/99	2

III.7. Algèbre relationnelle

CL >< CO

CL.NumCli	*Nom*	*CO.NumCli*	*Date*	*Quantité*
1	C1	1	22/09/99	1
3	C3	3	22/09/99	5
3	C3	3	22/09/99	2

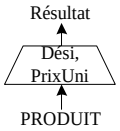
Nom	*Date*	*Quantité*
C1	22/09/99	1
C3	22/09/99	5
C3	22/09/99	2

Π <Nom, Date, Quantité> (CL >< CO)

(Projection sur les attributs Nom, Date, Quantité)

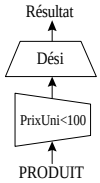
III.8. Exemples de requêtes

Ex. 1 : *Désignation et prix unitaire de tous les produits*



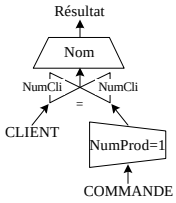
III.8. Exemples de requêtes

Ex. 2 : *Désignation des produits de prix inférieur à 100 €*



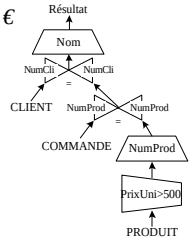
III.8. Exemples de requêtes

Ex. 3 : *Nom des clients qui ont commandé le produit n° 1*



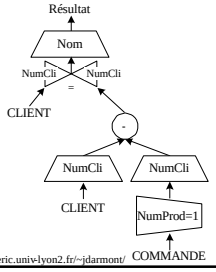
III.8. Exemples de requêtes

Ex. 4 : *Nom des clients qui ont commandé au moins un produit de prix supérieur à 500 €*



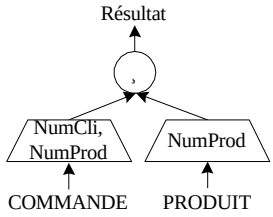
III.8. Exemples de requêtes

Ex. 5 : *Nom des clients qui n'ont pas commandé le produit n° 1*



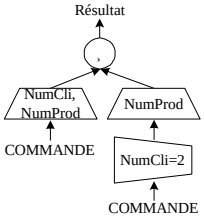
III.8. Exemples de requêtes

Ex. 6 : *Numéro des clients qui ont commandé tous les produits*



III.8. Exemples de requêtes

Ex. 7 : *Numéro des clients qui ont commandé tous les produits commandés par le client n° 2*



III.9. Classification des SGBD relationnels

- Niveau 1 : *Systèmes non relationnels.* Supportent uniquement la structure tabulaire.
- Niveau 2 : *Systèmes relationnellement minimaux*. Permettent les opérations de sélection, projection et jointure.
- Niveau 3 : *Systèmes relationnellement complets.* Toutes les opérations de l'algèbre relationnelle.
- Niveau 4 : *Systèmes relationnellement pleins*. Permettent la définition des contraintes d'intégrité.