Cours de Bases de données
F. Bannay

Table des matières

1 Introduction

Définition 1.1 *Un système de gestion de base de données (SGBD) est un ensemble de logiciels systèmes permettant aux utilisateurs* ***d'insérer, de modifier et de rechercher efficacement*** *des données dans une grande masse d'informations (milliards d'octets)* ***partagée par de multiples utilisateurs****.*

Cela suppose de :
- permettre recherche des données à partir d'une valeur ou de relations entre objets
- assurer le partage des données (chaque utilisateur se sente tout seul)
- protéger les données contre tout incident ou accident
- performances

Un SGBD est meilleur qu'un système de gestion de fichiers qu'il intègre, car :
- les données sont structurées : types, formats
- les opérations sont répertoriées : algèbre relationnelle

exemple d'applications :
- Gestion (salaires, stocks, ...)
- Transactionnel (comptes, centrales d'achat, ...)
- Réservations (avions, trains, ...)
- Librairie électronique (bibliothèques, journaux, web, ...)
- Documentation technique (nomenclature, plans, dessins,...)
- Bureautique (formulaires, textes, images, son, ...)
- Géographique (cartes routières, thématiques, ...)
- Génie Logiciel (programmes, manuels, ...)

2 Le modèle relationnel

Modèle relationnel (2ième génération) années 70: c'est le modèle standard utilisé dans les principaux systèmes : ORACLE, INGRES, SYBASE, INFORMIX, DB2, RDB.

Définition 2.1 (Relation) *Une relation est un* ensemble *de tuples (ou enregistrement ou n-uplets) de valeurs atomiques (insécables) prises dans des domaines. La cardinalité de la relation est le nombre de n-uplets dans cet ensemble.*

Comme c'est un ensemble,
- un tuple ne peut exister qu'une seule fois

– l'ordre dans lequel les tuples sont n'a pas d'importance

Valeur atomique : entier, chaîne de caractères, ...

ex : La relation ROBD(nom, prénom , auteur) (nom, prénom et auteur sont des attributs)

nom	prenom	auteur	
Gaston	Lagaffe	Franquin	← tuple
Triphon	Tournesol	Hergé	

Domaines :

D1 = { Gaston, Triphon, Lucie, Camille, ... }

D2 = { Lagaffe, Tournesol, Bannay, ... }

D3 = { Franquin, Hergé, ...}

Définition 2.2 (Attribut, tuple, schéma) *Un attribut est un nom donné à une colonne d'une relation.*
Un tuple est une ligne d'une relation. Il correspond à un enregistrement.
Le schéma d'une relation est le nom de la relation suivi de la liste des noms des attributs. Chaque attribut est associé à un domaine. Le degré d'une relation est son nombre d'attributs.

(C'est une définition de structure contrairement à la relation qui relie les valeurs)

ex : ROBD(nom, prénom, auteur) est une relation. (Gaston, Lagaffe, Franquin) est un tuple.

2.1 Opérations de l'algèbre relationnelle

Collection d'opérations formelles qui agissent sur des relations et produisent des relations. (Codd 1970).

2.1.1 Opérations ensemblistes

Les trois opérateurs ensemblistes opèrent sur des relations R et S de **même schéma** s_{RS}.

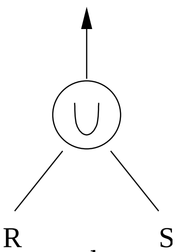
R :

nom	prenom
Dupont	Jean
Dubois	Jacques

S :

nom	prenom
Dupont	Jean
Block	Hélène
Petit	Jeannine

union : R ∪ S



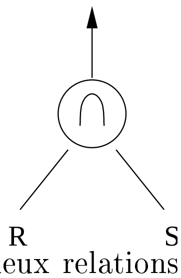
si R et S sont deux relations de même schéma alors R ∪ S produit une nouvelle relation de même schéma que R et S et ayant les tuples de R et ceux de S (les doublons sont supprimés).

sur l'exemple donne $R \cup S$:

nom	prenom
Dupont	Jean
Dubois	Jacques
Block	Hélène
Petit	Jeannine

Attention ! : on ne peut faire l'union de 2 relations que si elles ont même schéma.
Les relations dont on fait l'union peuvent être le résultat d'opérations de l'algèbre relationnelle.

intersection : $R \cap S$

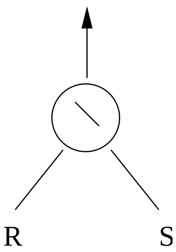


si R et S sont deux relations de même schéma s alors $R \cap S$ produit une nouvelle relation de schéma s et ayant les tuples présents à la fois dans R et dans S.

$R \cap S$:

nom	prenom
Dupont	Jean

différence : $R \setminus S$

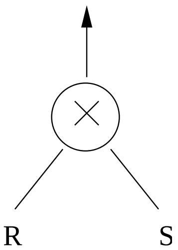


si R et S sont deux relations de même schéma s alors $R \setminus S$ produit une nouvelle relation de schéma s et ayant les tuples de R qui ne sont pas dans S. Attention, la différence n'est pas commutative.

$R \setminus S$:

nom	prenom
Dubois	Jacques

produit cartésien: (pas forcément même schéma) $R \times S$



si R et S sont deux relations de schéma r et s, $R \times S$ a un schéma t tq la liste des attributs de t est la liste des attributs de r suivie des attributs de s noté $r\hat{s}$ et dont les tuples sont toutes les combinaisons de la concaténation des tuples de R et de S.
Attention le nombre d'éléments du produit cartésien est le produit des cardinalités

des relations R et S!

R :

A1	A2
a	2
b	3

S :

A3	A4	A5
x	0	1
y	2	2
z	1	0

R x S

A1	A2	A3	A4	A5
a	2	x	0	1
a	2	y	2	2
a	2	z	1	0
b	3	x	0	1
b	3	y	2	2
b	3	z	1	0

2.1.2 Opérations spécifiques

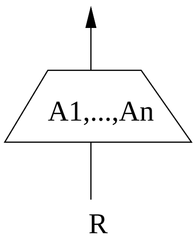
LesFilms

titre	pays	annee	realisateur	duree
La vie est belle	Italie	1998	Roberto Benigni	107
On connait la chanson	France	1998	Alain Resnais	120
La grande vadrouille	France	1966	Gérard Oury	125
Fantasia	USA	1940	Walt Disney	125

Roles

titrefilm	acteur
La grande vadrouille	De Funès
On connait la chanson	Agnès Jaoui
La grande vadrouille	Bourvil
On connait la chanson	Jean-Pierre Bacri
On connait la chanson	Sabine Azéma
On connait la chanson	Pierre Arditi
La vie est belle	Benigni

projection : $\pi_{\text{A1, ..., An}}$ (R)



Si R est une relation et A1, ... An sont des attributs pris parmis ceux de R alors $\pi_{\text{A1, ..., An}}$ (R) est la relation de schéma A1, ..., An (schéma inclus dans celui de R) ayant comme nuplets ceux de R restreints au sous-schéma A1, ..., An. Les tuples en doubles sont conservés en une seule fois.

Il faut noter que la cardinalité de la nouvelle relation est inférieure ou égale à celle de R, puisque des doublons ont pu être produits par la projection et sont donc supprimés (une relation est toujours un ensemble).

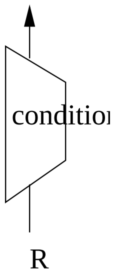
ex : $\pi_{titre,annee}$ (LesFilms)

titre	annee
La vie est belle	1998
On connait la chanson	1998
La grande vadrouille	1966

π_{annee} (LesFilms)

annee
1998
1966

restriction : σ_c (R)



Si R est une relation de schéma r et c une condition alors σ_c (R) est une relation de schéma r ayant comme nuplets ceux de R satisfaisant à une condition.
la condition c est

- soit une condition simple de la forme $\langle$ attribut opérateur valeur $\rangle$
- soit une combinaison de conditions (ou condition composée) de la forme
  - non cond1,
  - cond1 ou cond2,
  - cond1 et cond2

  où cond1 et cond2 sont des conditions (simples ou composées).

L'opérateur est un opérateur de comparaison choisi parmi $\{= < \leq \geq > \neq\}$. L'attribut doit appartenir à R
ex : Films de l'année 1998 :

$\sigma_{annee=1998}$(LesFilms)

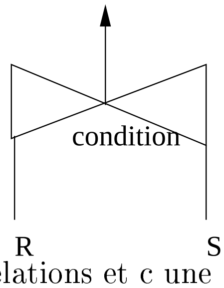
titre	pays	annee	realisateur	duree
La vie est belle	Italie	1998	Roberto Benigni	107
On connait la chanson	France	1998	Alain Resnais	120

ex : Films réalisés en France et de durée 120 :

$\sigma_{pays=\text{France et duree}\geq 120}$(LesFilms)

titre	pays	annee	realisateur	duree
On connait la chanson	France	1998	Alain Resnais	120
La grande vadrouille	France	1966	Gérard Oury	125

Jointure : $R \bowtie_{\text{condition}} S$



Si R et S sont deux relations et c une condition reliant des attributs de R et S alors $R \bowtie_c S$ est une relation de schéma r$\hat{\text{s}}$ (concaténation de r et de s) contenant l'ensemble de tous les tuples obtenus en concaténant un tuple de R et un tuple de S et vérifiant la condition de rapprochement.
condition de la forme Attribut1 opérateur Attribut2

Jointure naturelle quand condition est égalité (on enlève la dupplication de colonne) si on étend restriction à condition: attribut opérateur expression(attribut ou valeur...) alors restriction du produit cartésien.

Nom des acteurs et détails des films dans lesquels ils jouent.

Lesfilms $\bowtie$ Roles

titre	pays	annee	realisateur	duree	acteur
La vie est belle	Italie	1998	Roberto Benigni	107	Benigni
On connait la chanson	France	1998	Alain Resnais	120	Agnès Jaoui
On connait la chanson	France	1998	Alain Resnais	120	Jean-Pierre Bacri
On connait la chanson	France	1998	Alain Resnais	120	Sabine Azéma
On connait la chanson	France	1998	Alain Resnais	120	Pierre Arditi
La grande vadrouille	France	1966	Gérard Oury	125	De Funès
La grande vadrouille	France	1966	Gérard Oury	125	Bourvil

2.2 Théorème

Théorème 1 *L'algèbre relationnelle permet d'exprimer toute question exprimable avec la logique du premier ordre sans fonction.*

ex: $\{$x $\mid \exists f \in LesFilms$ et $\exists r \in Roles$ tq $f.titre = r.titrefilm$ et $x = r \,\hat{}\, f\}$ = LesFilms $\bowtie_{titre=titrefilm}$ Roles

Définition 2.3 (Arbre algébrique) *Arbre dont les feuilles sont des relations, les nœuds correspondent à des opérations de l'algèbre relationnelle.*

Films

Lesfilms(titre,pays,annee,realisateur,duree)
Roles(titrefilm,acteur)

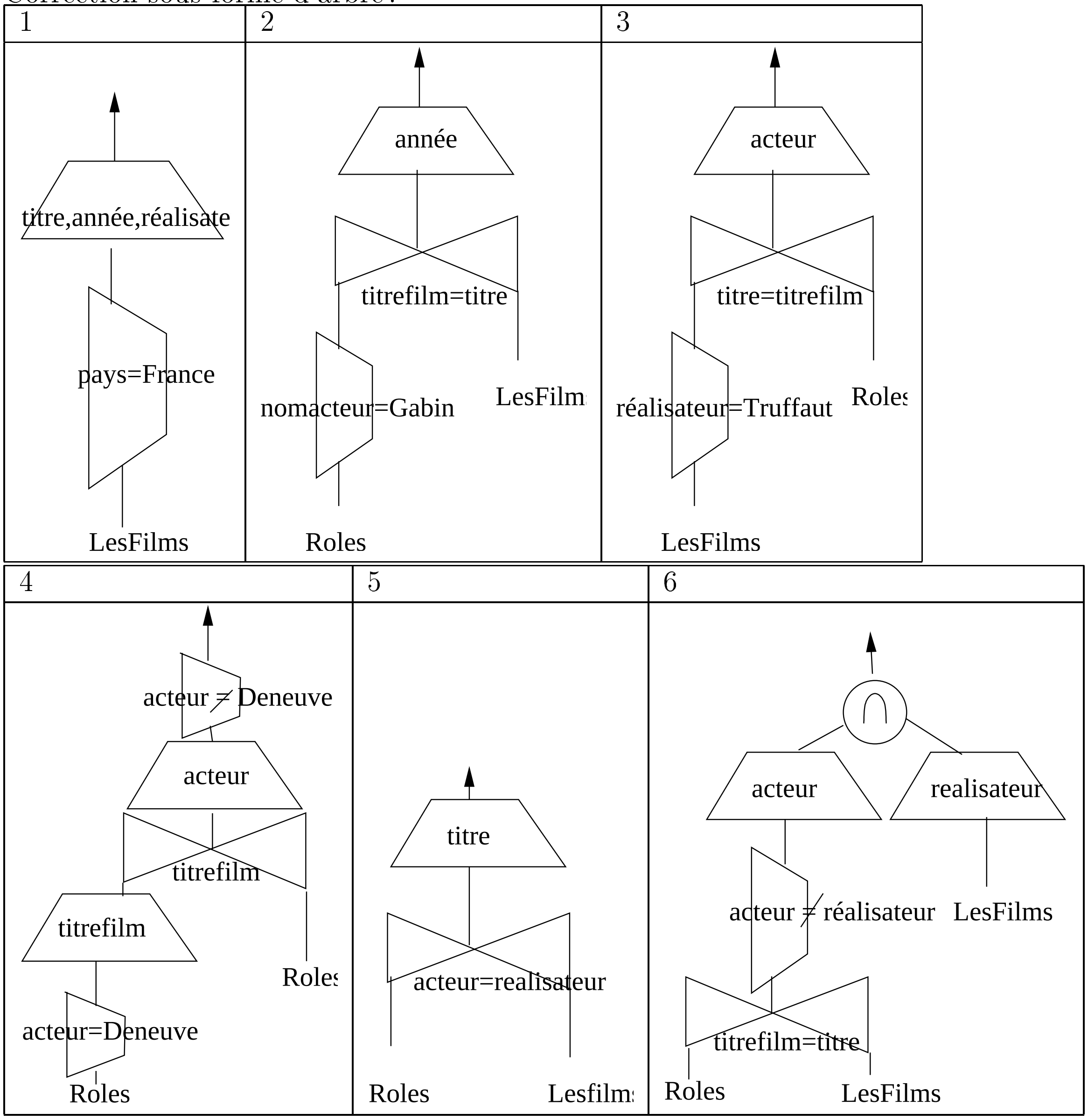
1. Liste des films français (titre,année,réalisateur)
2. Donnez les années de sortie des films tournés avec Gabin
3. Trouver les acteurs qui ont tourné avec TRUFFAUT comme réalisateur.
4. Trouver tous les acteurs qui ont été partenaires de DENEUVE.
5. Liste des films où le réalisateur est aussi acteur
6. Réalisateurs ayant joué comme acteurs dans des films qu'ils ne réalisaient pas eux-mêmes.
7. Réalisateurs qui ne jouent pas dans un de leur propre film.
8. Donnez les acteurs qui jouent dans tous les films de Truffaut. (Jean-Pierre Léaud)

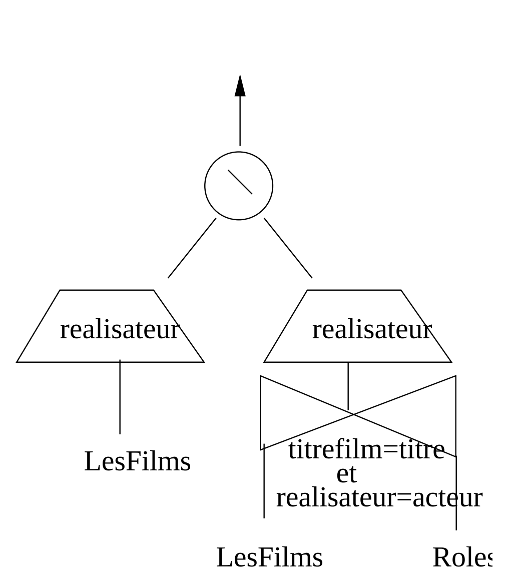
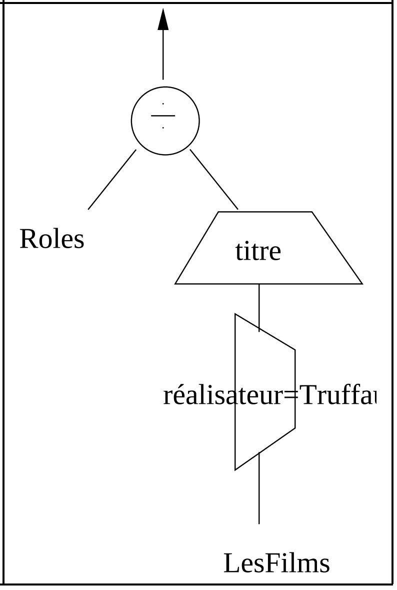
Correction sous forme de formule algébrique :

1. $\pi_{titre,annee,realisateur}(\sigma_{\text{pays=France}}(\text{Lesfilms}))$
2. $\pi_{annee}(\sigma_{acteur='Gabin'}(\text{Roles}) \bowtie_{titrefilm=titre} \text{Lesfilms}))$
3. $\pi_{acteur}(\sigma_{realisateur=Truffaut}(LesFilms) \bowtie_{titre=titrefilm} Roles)$

4. Films dans lesquels joue Deneuve
   $R1 = \pi_{titrefilm}(\sigma_{acteur=Deneuve}(Roles))$
   $\sigma_{acteur \neq Deneuve}(\pi_{acteur}(R1 \bowtie_{titrefilm} Roles))$
5. $\pi_{titre}(Roles \bowtie_{acteur=realisateur} LesFilms)$
6. acteurs qui ne sont pas réalisateurs du film $\cap$ réalisateurs
   $\pi_{acteur}(\sigma acteur \neq realisateur(Roles \bowtie_{titrefilm=titre} LesFilms) \cap \pi_{realisateur}(LesFilms)$
7. $\pi_{realisateur}(LesFilms) \setminus \pi_{realisateur}(LesFilms \bowtie_{\substack{titrefilm = titre \\ etrealisateur = acteur}} Roles)$
8. $Roles \div \pi_{titre}(\sigma_{realisateur=Truffaut}(LesFilms))$

Correction sous forme d'arbre :



7	8
	

2.3 Opérations dérivées

Intersection : Elle peut se définir à partir des autres opérateurs.
$R \cap S = R \setminus (R \setminus S) = S \setminus (S \setminus R)$

Complément : Si R est une relation de schéma A1, ..., An de domaines D1, ..., Dn alors le complement(R) est une relation de même schéma et dont les tuples sont ceux du produit cartésien des domaines mais qui n'appartiennent pas à R.
Complement(R) $= D1 \times D2 \times ... \times Dn \setminus R$
Si on ne connait pas le domaine on peut se baser sur ce que contient la table : Complement(R) $= \pi_{A1}(R) \times pi_{A2}(R) ... \times pi_{An}(R) \setminus R.$
ex : Complement(Roles) $=$ relie les acteurs avec les films dans lesquels ils ne jouent pas.
Complement(Roles)

titrefilm	acteur	titrefilm	acteur
La vie est belle	De Funès	La vie est belle	Sabine Azéma
On connait la chanson	De Funès	La grande vadrouille	Sabine Azéma
Fantasia	De Funès	Fantasia	Sabine Azéma
La vie est belle	Agnès Jaoui	La vie est belle	Pierre Arditi
La grande vadrouille	Agnès Jaoui	La grande vadrouille	Pierre Arditi
Fantasia	Agnès Jaoui	Fantasia	Pierre Arditi
La vie est belle	Jean-Pierre Bacri	On connait la chanson	Benigni
La grande vadrouille	Jean-Pierre Bacri	La grande vadrouille	Benigni
Fantasia	Jean-Pierre Bacri	Fantasia	Benigni

Df={La vie est belle, On connait la chanson, Fantasia, La grande vadrouille}
Da={De Funès, Agnès Jaoui, Jean-Pierre Bacri, Sabine Azéma, Pierre Arditi, Benigni}

Division R ÷ S : Si R et S sont deux relations de schémas respectifs A1, ...,Ap, Ap+1, ..., An et Ap+1,..., An alors R÷ S est la relation de schéma A1, ..., Ap contenant les tuples x tels que quel que soit y de S, xŷ appartient à R.

La division peut se définir à l'aide du produit cartésien, de la différence et de la projection :

R(A1, .., Ap, Ap+1, ..., An)

S(Ap+1, ..., An)

R ÷ S = $\pi_{A1,...Ap}(R) \setminus \pi_{A1,...Ap}(\pi_{A1,...Ap}(R) \times S \setminus R)$

ex : Quels sont les acteurs qui jouent dans tous les films. Soit F une table contenant la liste de tous les films :

F:

titre
La vie est belle
On connait la chanson
La grande vadrouille

R

titrefilm	acteur
La grande vadrouille	De Funès
On connait la chanson	De Funès
La grande vadrouille	Bourvil
On connait la chanson	Jean-Pierre Bacri
On connait la chanson	Sabine Azéma
On connait la chanson	Pierre Arditi
La vie est belle	De Funès

R ÷ F:

acteur
De Funès

acteurs qui ne jouent pas dans un film (complémentaire des acteurs qui jouent dans un film) : $\pi_{acteur}(R) \times F \setminus R$

acteurs qui jouent dans tous les films : $\pi_{acteur}(R) - \pi_{acteur}(\pi_{acteur}(R) \times F \setminus R)$

Jointure externe R $\bowtie_{condition}$ (+) S : c'est une jointure entre R et S qui permet d'obtenir les tuples formés à partir de R et de S vérifiant la condition auxquels sont ajoutés les tuples de R qui n'ont pas de correspondant à travers la condition

exemple : liste des films et de leurs acteurs (y compris les films sans acteurs)

Lesfilms $\bowtie_{titre=titrefilm}$ (+) Roles

titre	pays	annee	realisateur	duree	acteur
La vie est belle	Italie	1998	Roberto Benigni	107	Benigni
On connait la chanson	France	1998	Alain Resnais	120	Agnès Jaoui
On connait la chanson	France	1998	Alain Resnais	120	Jean-Pierre Bacri
On connait la chanson	France	1998	Alain Resnais	120	Sabine Azéma
On connait la chanson	France	1998	Alain Resnais	120	Pierre Arditi
La grande vadrouille	France	1966	Gérard Oury	125	De Funès
La grande vadrouille	France	1966	Gérard Oury	125	Bourvil
Fantasia	USA	1940	Walt Disney	125	

R $\bowtie_{Ai=Bj}$ (+) S = R $\bowtie_{Ai=Bj}$ S $\cup$ $\pi_{A1,...,An,Vide,Vide,...,Vide}((\pi_{Ai}(R) \setminus \pi_{Bj}(S)) \bowtie_{Ai} R)$

où Vide est une constante représentant la valeur vide d'un attribut. Vide est placé autant de fois qu'il y a d'attributs dans B.
(à expliquer après l'extension de la projection à une expression et non plus sur des attributs)

Semi jointure R $\ltimes_c$ S : C'est une jointure de R avec S dans laquelle on ne conserve que les colonnes de R.
R $\ltimes_c$ S $= \pi_r(R \bowtie_c S)$
exemple : Lesfilms $\ltimes$ Roles

titre	pays	annee	realisateur	duree
La vie est belle	Italie	1998	Roberto Benigni	107
On connait la chanson	France	1998	Alain Resnais	120
La grande vadrouille	France	1966	Gérard Oury	125

2.4 L'opération de Fermeture transitive (pas dans l'Algèbre Relationnelle)

Si R(A1,A2) est une relation à deux attributs ayant même domaine (domaine de A1 = domaine de A2) alors FT(R) est une relation ayant même schéma que R et pour tuples ceux de R auxquels on ajoute les tuples (a,c) tels qu'il existe b appartenant à D1 avec (a,b) appartient à FT(R) et (b,c) appartient FT(R).
Exemple : Gouvernement

nom	superieur	
Chirac		(président)
Raffarin	Chirac	(ministre)
Ferry	Raffarin	(ministre education)
Haignere	Ferry	(ministre recherche)
Bachelot	Raffarin	(ministre ecologie)

FT(Gouvernement)

nom	superieur
Chirac	
Raffarin	Chirac
Ferry	Raffarin
Ferry	Chirac
Haignere	Ferry
Haignere	Raffarin
Haignere	Chirac
Bachelot	Raffarin
Bachelot	Chirac

Exercices

Hopital

On considère la base de données suivante :
Personnel(idpers,nom,prenom,datnais,qualif,trmnt,cdsrv)
Service(cdsrv,nomserv,localisation)
Patient(idpat,nompat,prenom,profession,insee)
Consultation(idpat,idpers,date)

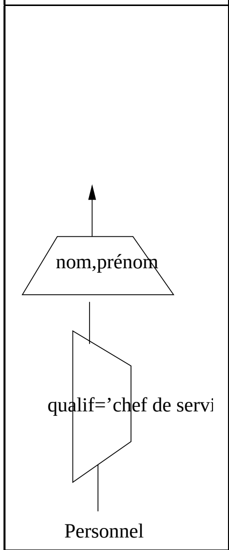
1. nom et prénom des personnels de qualification CHEF DE SERVICE
2. nom des infirmières de traitement supérieur à 8500F
3. nom du chef de service du service de Cardiologie
4. nom du service dans lequel le patient Dupont a consulté le 3 juillet 2001
5. nom des patients dont une consultation est en Cardiologie
6. nom des patients dont toutes les consultations sont en Cardiologie

Correction sous forme de formule algébrique:

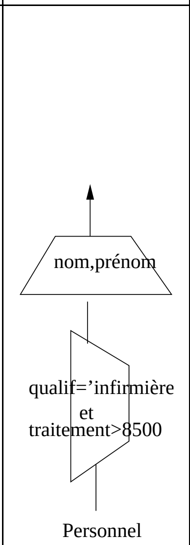
1. $\pi_{nom,prenom}(\sigma_{qualif='CHEFDESERVICE'}(Personnel))$
2. $\pi_{nom}(\sigma_{qualif=\text{'Infirmière' et } trtmnt>8500}(Personnel))$
3. $\pi_{nom}(\sigma_{qualif='CHEFDESERVICE'}(Personnel) \bowtie_{cdsrv} \sigma nomserv =' \, Cardiologie'(Service))$
4. $\pi_{nomserv}(Service \bowtie_{cdsrv} (Personnel \bowtie_{idpers} (\sigma_{date='3-jul-2001'}(Consultation) \bowtie_{idpat} (\sigma_{nompat='Dupont'}(Patient)))))$
5. $\pi_{nompat}(Patient \bowtie_{idpat} (Consultation \bowtie_{idpers} (Personnel \bowtie_{cdsrv} \sigma_{nomserv='Cardiologie'}(Service))))$
   ou $\pi_{nompat}((Patient \bowtie_{idpat} Consultation) \bowtie_{idpers} (Personnel \bowtie_{cdsrv} \sigma_{nomserv='Cardiologie'}(Service)))$
6. $\pi_{nompat}(Patient) \setminus$ 5 avec $\neq$ 'Cardiologie'

Correction sous forme d'arbre algébrique :

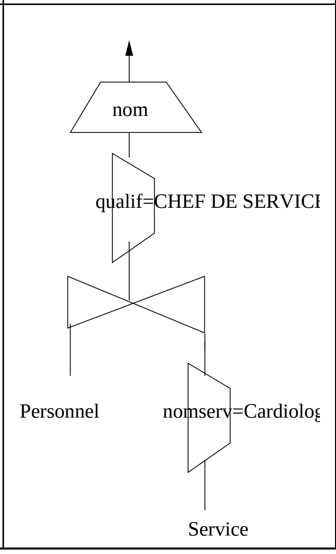
1



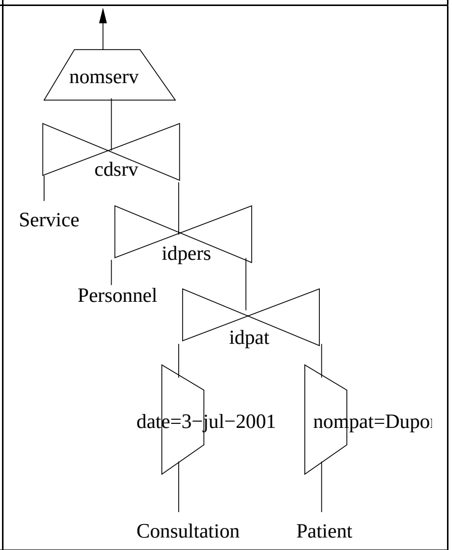
2

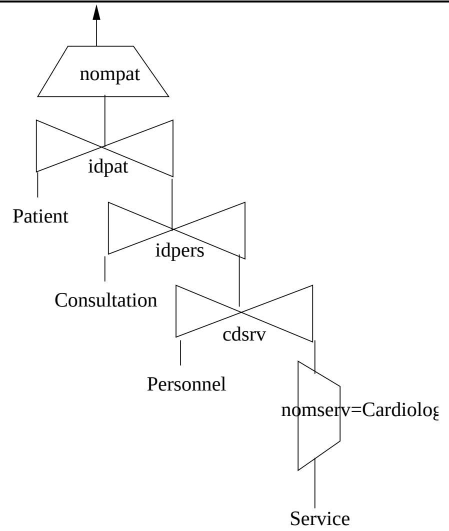
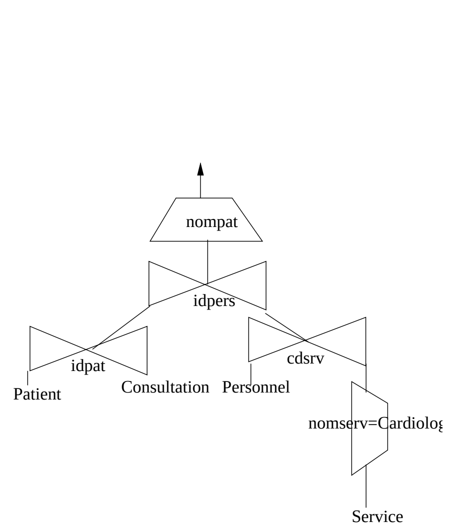
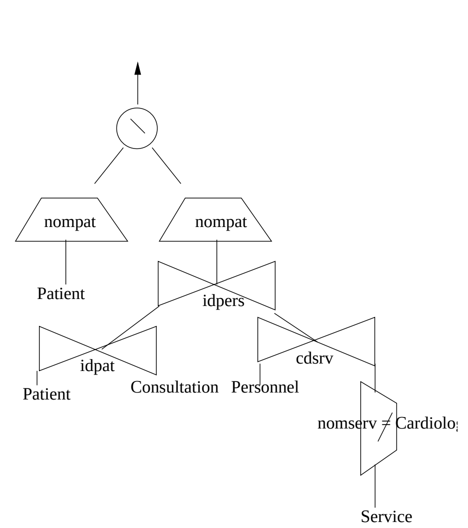


3



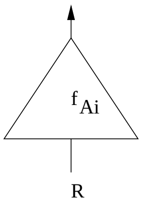
4

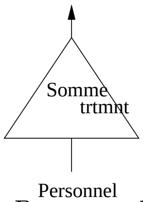


5	ou encore	6
		

2.5 Extension de l'algèbre relationnelle : Fonctions et agrégats

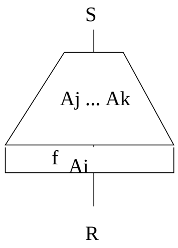
Définition 2.4 *On appelle fonctions de groupe les fonctions suivantes : Somme, Compte, Moyenne, Min et Max.*
On appelle agrégation élémentaire*, une opération qui à partir d'une relation R, d'un attribut A_i et d'une fonction de groupe f associe* une seule valeur*, calculée en appliquant la fonction de groupe f à tous les éléments de la colonne A_i de la relation R. Notée $f_{A_i}(R)$*

ou graphiquement : 

Exemple 2.1 $Somme_{trmnt}(Personnel)$ *ou* 

renvoie la somme des salaires de la relation Personnel. C'est un nombre.

Définition 2.5 *On appelle* agrégation ensembliste*, une opération qui à partir d'une relation $R(A_1, \ldots A_n)$, d'un attribut A_i, d'une fonction de groupe f et d'un ensemble d'attributs $A_j, \ldots A_k$ de R permet d'obtenir la relation :*

$S(A_j,\ldots,A_k,f) = f_{A_i}([A_j \ldots A_k],R)$ représentée par : 

telle que
$\forall x \in R, \exists y \in S$ *tel que* $y.A_j = x.A_j \ \ldots \ y.A_k = x.A_k$.
et
$\forall y \in S, y.f = f_{x \in R, x.A_j = y.A_j \ldots x.A_k = y.A_k} x.A_i$

Exemple 2.2 *$Moyenne_{trmnt}([Qualif],Personnel)$ renvoie la table*

$Moyenne_{trmnt}([qualif],Personnel)$	
QUALIF	*Moyenne(TRTMNT)*
AIDE SOIGNANTE	*5020.25*
ASSISTANT	*25368.8625*
CHEF DE CLINIQUE	*22822.9333*
CHEF DE SERVICE	*34644.35*
CHEF INFIRMIERE	*11390.45*
DOCTEUR	
INFIRMIERE	*8152.53333*
INTERNE	*10030.7667*
MANIPULATEUR RADIO	*8500.15*
SECRETAIRE MEDICALE	*5245.65*

exercice :
Moyenne des salaires par service et par qualification.
$Moyenne_{trmnt}([codeserv,qualif],Personnel)$

codeserv	*qualif*	*moyenne*
01	*ASSISTANT*	*21327.35*
01	*CHEF DE SERVICE*	*30346.75*
01	*INTERNE*	*9678.25*
01	*SECRETAIRE MEDICALE*	*4830.45*
02	*AIDE SOIGNANTE*	*5020.25*
02	*ASSISTANT*	*27364.5*
02	*CHEF DE SERVICE*	*35764.25*
02	*INFIRMIERE*	*8218.15*
02	*INTERNE*	*10320.3*
02	*SECRETAIRE MEDICALE*	*5720.85*

Remarque 1 Les fonctions de groupe ne tiennent pas compte des valeurs vides, par exemple $Compte_{A_i}(R)$ renvoie le nombre d'éléments non VIDE dans la colonne A_i.

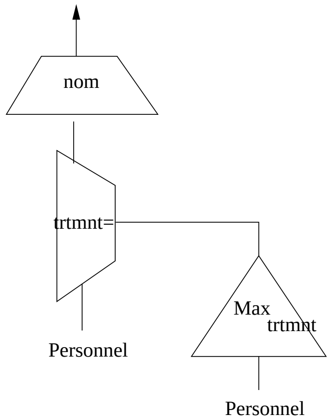
Définition 2.6 *Extension de l'algèbre relationnelle :*

– *Les* agrégations ensemblistes *sont utilisables dans les requêtes au même titre que les* *opérations ensemblistes* $\cup$, $\cap$, $\setminus$ *et que les opérations spécifiques* Π, σ, $\bowtie$.

- *Tout attribut utilisé dans une opération peut être remplacé par une* expression d'attributs *utilisant*
  - *soit des opérateurs arithmétiques :* +, -, *, /, ...
  - *soit des opérateurs sur les chaînes : concaténation (notée ||), extraction de sous-chaîne (notée sschaine(c,i,n) = n caractères de la chaîne c à partir du caractère i), ...*
- *Toute constante utilisée dans une expression ou dans une condition peut être remplacée par une* agrégation élémentaire.

Exemple 2.3

- $\sigma_{nom=Dupontetprenom=titi}(Personnel)\ LeftRightarrow\ \sigma_{nom||prenom=Duponttiti}(Personnel)$
- *nom de la personne la mieux payée de l'hopital:*
  $\pi_{nom}(\sigma_{traitement=Max_{traitement}(Personnel)}(Personnel))$.



Exercices :

1. Employés dont le nom commence par un 'A'.
2. Moyenne des salaires de la table Personnel sans tenir compte des salaires minimaux et maximaux.
3. Numéro id de la personne accompagné de son revenu en Euro.
4. Nombre de personnes par service. (codeserv, nbre de pers)
5. Nombre d'infirmière par service (nom du service, nombre d'infirmières). Ceci pour tous les services même s'ils n'ont pas de personnel.

Correction :

1. $\sigma_{sschaine(nom,1,1)='A'}(Personnel)$
2. $Moyenne_{trtmnt}(\sigma_{trmnt \neq Min_{trmnt}(Personnel)\text{ et }trmnt \neq Max_{trmnt}(Personnel)}(Personnel)$
3. $\Pi_{idpers,(trmnt+vctn)/6.559}(Personnel)$
4. $Compte_{idpers}([cdsrv],Personnel)$
5. $Compte_{idpers}([cdsrv,nomserv],Service \bowtie_{cdsrv} (+)\sigma_{qualif=infirmiere}(Personnel))$

3 Langage SQL

C'est le langage utilisé dans toutes les BD relationnelles. (Structured Query Language). C'est un standard.
LDD: Langage de definition des données (structure des relations) LMD: langage de manipulation des données LID: langage d'interrogation des données.

3.1 Interrogation

```
SELECT [DISTINCT|ALL] liste expressions
FROM liste tables
[WHERE condition]
[GROUP BY liste attributs [HAVING condition]]
[ORDER BY liste expression]
[<UNION | UNION ALL | INTERSECT | MINUS>
 instruction select ]
```

Propriété 3.1 *La commande SELECT permet d'exprimer toute question exprimable en AR.*

3.1.1 Les opérateurs de l'algèbre relationnelle classique

- projection $\pi_{\text{Ai, ..., Ak}}$ sachant que R(A1,..An).

  `SELECT DISTINCT Ai, ..., Ak FROM R`

  ex :

  `SELECT DISTINCT annee FROM LesFilms`

  Remarque 2 La projection peut se faire sur toute expression formée à partir des attributs de R et de constantes.
  ex : $\pi_{nom,trtmnt/6.559}(Personnel)$

  `SELECT DISTINCT nom, trtmnt/6.559 FROM Personnel`

  ex :

  `SELECT DISTINCT 5*2+4.5 FROM Personnel`

  =>14,5
  Par défaut, SELECT renvoie tous les tuples de la table même les doublons, c'est-à-dire que SELECT = SELECT ALL.
- restriction $\sigma_{condition}(R)$.

  `SELECT * FROM R WHERE condition`

  où condition est une expression booléenne qui porte sur des attributs de R.
  ex : infirmières de salaire >8500F

  `SELECT * FROM Personnel WHERE trtmnt > 8500 AND qualif='infirmiere'`

- union, intersection, différence sur R et S de même schéma
  - `(SELECT * FROM R) UNION (SELECT * FROM S)`
  - `INTERSECTION, MINUS`
  - `UNION ALL`

    : si $x \in R$ et $x \in S$ alors il apparaîtra 2 fois dans UNION ALL.
- produit cartésien $R \times S$

  `SELECT * FROM R,S`
- jointure $R \bowtie_{R.Ai=S.Bj} S$ C'est un produit cartésien suivi d'une restriction.

  `SELECT * FROM R,S WHERE R.Ai=S.Bj`

  ex: Nom de la personne, sa profession, le nom du service dans lequel elle travaille.
  $\pi_{nom,qualif,nomserv}(Service \bowtie_{codeserv} Personnel)$

  ```
  SELECT nom,qualif,nomserv FROM Service,Personnel
  WHERE Service.codeserv=Personnel.codeserv
  ```
- jointure externe ex : nom de la personne, sa profession, le nom du service dans lequel elle travaille mais avec tous les services même ceux qui n'ont pas de personnel.

  ```
  SELECT nom,qualif,nomserv FROM Service,Personnel
  WHERE Service.codeserv=Personnel.codeserv (+)
  ```

3.1.2 Les agrégats

Les fonctions de groupe : SUM, COUNT, AVG, MAX, MIN, VARIANCE, s'utilisent soit pour obtenir des agrégations élémentaires soit pour obtenir des agrégations ensemblistes.

- agrégation élémentaire $f_{Ai}(R)$ applique la fonction de groupe f sur toute la colonne Ai de R.

  ex: somme des salaires du personnel

  `SELECT SUM(trtmnt) FROM Personnel`

  renvoie une seule ligne.
- agrégation ensembliste : renvoie une table ayant pour colonnes des colonnes contenant les attributs regroupés et une colonne contenant le résultat de la fonction appliquée à chacun de ces groupes.

  ```
  GROUP BY liste_attributs [HAVING c]
  ```

  exemple : Moyenne des salaires par qualification.

  ```
  SELECT qualif, avg(trtmnt)
    FROM personnel
    GROUP BY qualif;

  QUALIF                    AVG(TRTMNT)
  ```

```
------------------------ -----------
AIDE SOIGNANTE                 5020.25
ASSISTANT                   25368.8625
CHEF DE CLINIQUE            22822.9333
CHEF DE SERVICE               34644.35
CHEF INFIRMIERE               11390.45
DOCTEUR
INFIRMIERE                  8152.53333
INTERNE                     10030.7667
MANIPULATEUR RADIO             8500.15
SECRETAIRE MEDICALE            5245.65

10 rows selected.
```

Si une fonction de groupe est utilisée (SUM,AVG,...), il faut que dans le SELECT les seuls attributs qui apparaissent soient cités dans un GROUP BY.

ex : code du service, nom du service avec moyenne des traitements

```
SELECT codeserv,nomserv, avg(trtmnt)
  FROM personnel,service
  WHERE personnel.codeserv=service.codeserv
  GROUP BY codeserv,nomserv
```

Quand on utilise un GROUP BY on peut utiliser l'instruction HAVING qui permet de spécifier une condition portant sur chaque groupe.
ex : nom des services avec moyenne des salaires pour tous les services dont le nom commence par 'A'

```
SELECT service.nom, avg(trtmnt)
  FROM personnel,service
  WHERE personnel.cdsrv=service.cdsrv
  GROUP BY service.nom
  HAVING service.nom like 'R%'
```

cette question peut se résoudre ainsi :

```
SELECT service.nom, avg(trtmnt)
  FROM personnel,service
  WHERE personnel.cdsrv=service.cdsrv
  AND service.nom like 'A%'
  GROUP BY service.nom
```

ex : (celle-ci nécessite un HAVING) le service qui a le plus d'employés, ou bien si on veut un nombre d'employés supérieur à 10.

```
SELECT nomserv, count(idpers)
  FROM personnel,service
  WHERE personnel.codeserv=service.codeserv
```

```
GROUP BY nomserv
HAVING count(idpers)=Max(count(idpers))
```

3.1.3 Au delà de l'algèbre relationnelle

Remarque : Le langage SQL est plus expressif que l'AR

3.1.3.1 tri

ORDER BY liste de trucs ASC ou DESC
truc= position ou expression d'attributs et d'alias de colonnes,
ex: employés triés par salaire décroissant et ordre alphabétique du nom.

```
SELECT * FROM personnel ORDER BY traitement DESC, nom ASC.
```

Par défaut, le tri est ascendant.

3.1.3.2 requêtes hiérarchiques

Ne peut être utilisé que sur une seule table dans le FROM.

```
SELECT ...
FROM ...
WHERE ...
[START WITH condition]
[CONNECT BY condition]
```

ex: Personnes(idpers,nom,prenom,qualif,traitement,datnais,codesup)

idpers	nom	prenom	qualif	traitement	datnais	codesup
1	Chirac	Jacques	président			
2	Raffarin	JP	ministre			1
3	Ferry	Luc	ministre education			2
4	Haignere	Claudie	ministre recherche			3
5	Bachelot	Roselyne	ministre ecologie			2

START WITH : précise quelles sont les racines de l'arbre CONNECT BY : précise le lien entre un tuple père et son fils
ex :

```
SELECT nom, qualif FROM Personnes
START WITH codesup IS NULL
CONNECT BY codesup = PRIOR idpers
```

J'écris un tuple si son codesup est l'idpers du tuple précédent.
Sans le START WITH tous les tuples de la table initiale sont considérés successivement comme racines de l'arbre.
exercice : faire un requête qui renvoie la fermeture transitive de Personnes.

```
SELECT DISTINCT a.nom, b.nomsup
FROM superieur a, superieur b
WHERE b.nomsup in (SELECT DISTINCT c.nomsup from superieur c
                   START WITH c.nom=a.nom
```

```
CONNECT BY c.nom = prior c.nomsup)
```

On considère la table **Segment** suivante :

origine	destination	longueur
Angers	Paris	350
Boulogne	Dunkerque	100
Nantes	Angers	100
Paris	Boulogne	200
Paris	Strasbourg	500

1. Faîtes une requêtes permettant d'afficher toutes les destinations atteignables depuis Angers (sur l'exemple on obtient : Paris, Boulogne, Strasbourg, Dunkerque).
2. Faîtes une requête permettant d'afficher la fermeture transitive de la table Segment (i.e. pour toute les origines de la table segment il faut afficher les destinations atteignables) :

origine	destination
Angers	Boulogne
Angers	Dunkerque
Angers	Paris
Angers	Strasbourg
Boulogne	Dunkerque
Nantes	Angers
Nantes	Boulogne
Nantes	Dunkerque
Nantes	Paris
Nantes	Strasbourg
Paris	Boulogne
Paris	Dunkerque
Paris	Strasbourg

```
select s.destination
from lessegments s
start with s.origine='Angers'
connect by s.origine= prior s.destination;
```

si on ne met pas le `and prior s.destination!='Angers'`... il y a une erreur : `CONNECT BY loop in user data`

```
select distinct a.origine, b.destination
from lessegments a, lessegments b
where b.destination in (select distinct c.destination from lessegments c
start with c.origine=a.origine
connect by c.origine=prior c.destination);
```

Remarque : c'est bien au delà de l'AR car l'opération de fermeture transitive ne fait pas partie de l'AR.

3.2 précisions sur la syntaxe :

- renommage des colonnes : cf. TP
- renommage des tables : on peut placer des surnoms après les tables dans le FROM
- expressions : booléennes, arithmétiques, chaines
  - conditions : AND OR NOT
  - comparaisons arithmétiques : >,<,=,<=,>=,!=, between, not between ex : age >=35, age between 20 40, age >=35 and length(nom)<=5
  - comparaisons de chaine : =, like masque (%,_)
  - appartenance : IN, NOT IN : attribut [NOT] IN <exp| (SELECT ... FROM ... WHERE...)>ex : cours IN ('CHIMIE','PHYSIQUE') cours IN (SELECT cours FROM Service WHERE Q)
    - comparaison par rapport au résultat d'une requête si la requête ne renvoie qu'une seule valeur
    - association d'un comparateur avec ANY(au moins une) ALL(toutes) paye >= ANY(SELECT paye FROM salaire)
  - teste si case vide : attribut IS [NOT] NULL

  NOT EXISTS (SELECT ....) : vrai si ensemble non vide.
- cas particulier : autojointure, count(*) nombre de lignes même vides

3.2.1 Les fonctions standards

3.2.1.1 les fonctions numériques

- ABS (n), COS(n), SIN(n), EXP(n), LN(n), SQRT(n)
- CEIL (n) : le plus petit entier >= n
- FLOOR (n) : le plus grand entier <= n
- MOD (m,n) : le reste de la division entière de m par n
- POWER (m,n) : m puissance n
- LOG (m,n) : logarithme en base m de n
- ROUND (n,m) : la valeur de n arrondie à m positions à droite si m est positif, à gauche sinon
- TRUNC (n,m) : tronque au lieu d'arrondir
- SIGN (n) : -1 si n <0, 0 si n = 0, +1 si n >0

3.2.1.2 les fonctions sur les chaines

- CHR (n) : caractère de code ASCII n
- ASCII (car) : le code ascii de car (ou du premier caractère de car)
- LOWER (ch) : met la chaîne ch en minuscules
- UPPER (ch) : met la chaîne ch en majuscules
- INITCAP

- LENGTH (ch) : longueur de la chaîne ch
- SUBSTR (ch,pos,[long]) : extrait de la chaîne ch une sous-chaîne de longueur long commençant en position pos. Si pos est négatif la position est comptée à partir de la droite de ch.
- INSTR
- RTRIM (ch,' ') : enlève les espaces à droite de la chaîne, etc.
- LTRIM, RPAD, LPAD
- CONCAT ou ||
- REPLACE,
- SOUNDEX,
- TRANSLATE

3.2.1.3 les fonctions sur les dates

- SYSDATE : date et heure courantes du système
- MONTHS-BETWEEN (d1,d2) : nombre de mois entre 2 dates
- ADD_MONTHS
- LAST_DAY
- NEXT_DAY ex :

  ```
  select next_day(sysdate,'tuesday') from dual;

  NEXT_DAY(
  ---------
  27-NOV-01
  ```

- conversions
  de date vers chaîne : TO_CHAR (date, format)
  de chaîne vers date : TO_DATE (chaîne, format)

  On trouve comme format de date (avec un séparateur au choix), par exemple :

  ```
               'dd-mon-yyyy' => 26-NOV-1999
               'dd month yyyy' => 26 NOVEMBER 1999
               'dd/mm/yy' => 26/11/99
               'dd/mm/yyyy' => 26/11/1999
         day : tuesday
         cc : 21
         yyyy:2001
         yyy: 001
         yy:01
          y:1
        year :two thousand one
  ```

```
q:4 (trimestre “quarter”)
month : november
mon:nov
mm : 11
rm :ix
ww: 47 (numéro de la semaine dans l’année)
 w : 3 (numéro de la semaine dans le mois)
ddd:324 (numéro du jour dans l’année)
dd:20 (numéro du jour dans le mois)
dy:tue
d: 3 (numéro du jour de la semaine)
hh: 2 (heure sur 12)
hh24 : 14 (heure sur 24)
mi : 28 (minutes)
```

```
select to_char(to_date('08-jan-2002'),'w') from dual;
2
select to_char(sysdate,'RM:ww:w:ddd:dd:j:day:dy:d:hh12:hh24:mi') from dual;
```

3.2.1.4 autres fonctions

- NVL (exp1, exp2) : retourne exp2 si exp1 est NULL, exp1 sinon. Par exemple, on peut s’en servir pour remplacer NULL par des étoiles à l’affichage

  ```
  SELECT NVL (nom_col,'*****') ...
  ```

  ou pour éliminer les occurrences ayant 0 et garder celles qui ont NULL dans une colonne de type numérique en utilisant la condition :

  ```
  WHERE NVL (nom_col,1) <> 0
  ```

- DECODE (expression, valeur1, résultat1, valeur2, résultat2 ...) : retourne résultat1 si expression = valeur1 résultat2 si expression = valeur2, etc.

  Par exemple, on peut s’en servir pour exprimer un résultat dépendant d’un seuil de température

  ```
  SELECT DECODE (SIGN(temperature-seuil), -1, résultat1, 0, résultat2, 1, résultat2)
     FROM nom_table
     WHERE ... ;
  ```

- conversion de chaîne vers nombre : TO_NUMBER (chaîne, format)
- GREATEST, LEAST
- USER : nom de l’utilisateur

3.2.2 Les commandes sqlplus

SQLPLUS = Interface permettant de faire executer des commandes SQL.

```
HELP
```

```
START nomfichier ou @nomfichier
ACCEPT nomdevariable
PROMPT texte
```

Pour qu'une requête soit paramétrée, il faut placer &nom_de_variable dans la requête SQL.
Pour modifier l'affichage d'une requête, on peut utiliser

```
COL nomdecol FORMAT masque
COMPUTE operation OF nomcol ON elementderupture
```

3.3 Définition et Manipulation des données

Sous Oracle, en SQL2, on a accès (seulement) aux types suivants :

CHAR(taille) : chaîne de caractères (taille : défaut 1, max 255) (ex: 'toto')

VARCHAR(taille) : chaîne de caractères de longueur variable (1, 2000)

LONG : idem mais jusqu'à 2 gigaoctets

RAW et LONG RAW : données binaires (taille idem que varchar et long)

NUMBER(p, s) : nombre total de chiffres (précision) p(1 à 38) et nombre de chiffre après la virgule (échelle) s (-84 à 127)

ROWID : valeurs de la pseudo-colonne ROWID (adresse physique de chaque enregistrement)

DATE : date entre le 1er janvier -4712 et le 31 décembre 4712

Remarque le type BOOLEAN n'est pas un type SQL, mais c'est un type PL/SQL.

3.3.1 Définition et manipulation de schémas

création, destruction, modification du schéma d'une table : cf. commandes CREATE TABLE, DROP TABLE et ALTER TABLE.
cas particulier création d'une table à partir d'une requête : CREATE TABLE toto AS (SELECT ...); crée la table avec les données dedans

3.3.2 Manipulations des données

insertion, suppression, modification de données dans une table : cf. commandes INSERT, DELETE, UPDATE
cas particulier: INSERT INTO Personnel(nom,prenom) VALUES('Dupont','Jacques') Les autres attributs auront la valeur NULL.
INSERT INTO Personnel SELECT...
DELETE FROM nomdelatable [WHERE condition] : efface tous les tuples [qui vérifient la condition]

```
UPDATE nomdelatable
SET nomattribut= valeur ou requête
WHERE condition
```

modifie la valeur de l'attribut pour tous les tuples qui vérifient la condition.

3.4 Les Vues

Contrairement à une table (réelle, occupant un espace physique propre), une vue est une table virtuelle sans existence propre. Une vue est dynamique et se différencie en cela d'une table temporaire statique qui serait créée à partir d'autres tables existantes, elle existe encore si l'on supprime les tables dont elle est issue.
Une vue représente à tout instant le contenu exact des tables de la base. Toute modification effectuée sur celle-ci est instantanément répercutée sur la vue, et si l'une des tables de base est supprimée, la vue l'est aussi.
CREATE VIEW toto AS SELECT ...
Remarques : On ne peut pas mettre de ORDER BY lors de la création d'une vue (ni d'une table), on peut créer une vue à partir d'autres vues.
ex:

```
CREATE VIEW cardiologie AS
SELECT cdprs,nom, prnm, datnais
FROM personnel
WHERE cdsrv=(SELECT cdsrv FROM service WHERE nom='cardiologie')
```

Une vue est interrogeable comme une table.

```
SELECT * FROM cardiologie;
INSERT INTO Personnel VALUES('01','Titi','...','07',...);
SELECT * FROM personnel;
SELECT * FROM cardiologie;
```

où 07 est le code du service de cardiologie, dans les deux cas la personne Titi apparaît.

3.5 Les Séquences

Une séquence est une fonction particulière permettant de récupérer des valeurs séquentielles grace à la pseudo colonne NEXTVAL (si on ne veut pas incrémenter on peut utiliser CURRVAL)
ex :

```
create sequence toto start with 1;
select toto.Nextval, nom, prnm from personnel;
 NEXTVAL NOM             PRNM
---------- --------------- ---------------
         1 NOTTRET         Patrice
         2 CHOUBAT         Georges
         3 GADRET          Bernard
   ..
        38 GUYOT           Catherine
```

```
        39 BERTRAND        Christine
        40 MARTIGNY        Gisele

40 rows selected.
```

```
select toto.currval from dual;
   CURRVAL
----------
        40

1 row selected.
```

```
drop sequence toto;
```

Remarque si on veut un numéro associé à un tuple de requête on peut utiliser la pseudo colonne ROWNUM

```
select rownum, nom, prnm from personnel where rownum<5;
   ROWNUM NOM             PRNM
---------- --------------- ---------------
         1 NOTTRET         Patrice
         2 CHOUBAT         Georges
         3 GADRET          Bernard
         4 BISTEUR         Jules

4 rows selected.
```

3.6 Les Réflexes

Un réflexe est une opération sur la base de donénes qui permet de répercuter des mises à jour afin par exemple de gérer des données redondantes.

Définition 3.1 *Un réflex ou Trigger est un événement déclenché à l'apparition d'une condition particulière dans la base de données provoquant l'exécution d'un programme de traitement de la base composé d'une ou plusieurs requêtes*

```
CREATE OR REPLACE TRIGGER t_sup_srv_prs
       BEFORE DELETE ON service
              FOR EACH ROW
       WHEN ( :OLD.nom !='CARDIOLOGIE' )
       BEGIN
            DELETE FROM personnel
                   WHERE cdsrv = :OLD.cdsrv;
       END;
```

Ce réflexe détruit les personnes du service qui va être détruit sauf pour le service de cardiologie.

4 DF, clés, Normes (à passer)

La normalisation permet de définir des principes sur les relations à manipuler, afin d'éviter les redondances et les anomalies dans la manipulation des données de ces relations.
exemple : Approvisionnement

Produit	Quantité	Couleur	Fournisseur	Adresse
parapluie	110	rouge	Labaleine	Paris
chapeau	50	vert	Lemelon	Lyon
sac à main	65	noir	Toutcuir	Lyon
parasol	15	jaune	Labaleine	Paris
ombrelle	5	rouge	Labaleine	Paris
ceinture	25	vert	Letour	Nantes
sac à main	65	noir	Legrand	Paris

Solutions :
- étude des dépendances entre données
- décomposition et normalisation des relations

4.1 Dépendances fonctionnelles

Définition 4.1 *Etant donnés deux attributs X et Y d'une relation R, il y a une dépendance fonctionnelle entre X et Y (notée $X \to Y$) si la connaissance de la valeur de X détermine la valeur de Y de façon unique.*

ex: Fournisseur $\to$ Adresse.
Propriétés des dépendances fonctionnelles :

1. réflexivité : $Y \subseteq X \Rightarrow X \to Y$
2. augmentation : $X \to Y \Rightarrow X,Z \to Y,Z$
3. transitivité : $X \to Y$ et $Y \to Z \Rightarrow X \to Z$
4. pseudo-transitivité : $X \to Y$ et $Y,W \to Z \Rightarrow X,W \to Z$
5. union : $X \to Y$ et $X \to Z \Rightarrow X \to Y,Z$
6. décomposition : $X \to Y,Z \Rightarrow X \to Y$ et $X \to Z$

Définition 4.2 (dépendance élémentaire) *Une df $A \to B$ est* élémentaire *ou* pleine *s'il n'existe pas $A' \subseteq A$ tel que $A' \to B$*

Exemple : Fournisseur, Produit $\to$ Adresse

Définition 4.3 (dépendance directe) *Une df $A \to B$ est* directe *s'il n'existe pas C tel que $A \to C$ et $C \to B$ (c'est-à-dire non déduite par transitivité).*

exemple : Fournisseur $\to$ codepostal n'est pas directe car Fournisseur $\to$ Adresse et Adresse $\to$ codepostal

Définition 4.4 (clé) *La relation R(A1,...,An) admet pour clé le groupe d'attribut (A1, ...,Ak) si (A1,...Ak) $\to$ Ak+1, ...An (si la valeur des attributs A1, ..., Ak détermine la valeur de tous les autres attributs)*

Remarque : une relation peut avoir plusieurs clés, une clé peut ne contenir qu'un seul attribut.

4.2 formes normales

1FN : une relation est en première forme normale ssi tout attribut a une valeur atomique (vrai par définition du modèle relationnel)

2FN : une relation est en deuxième forme normale ssi elle est en 1FN et si tous les attributs non clés sont pleinement dépendants des clés (toutes les dépendance entre une clé et un attribut non clé sont élémentaires)

3FN : une relation est en troisième forme normale ssi elle est en 2FN et si tous les attributs non clés sont directement et pleinement dépendants des clés (si tout attribut non clé ne dépend pas d'un autre attribut non clé)

FNBC (Boyce-Codd) : une relation est en FNBC ssi elle est en 3FN et si tout attribut non clé n'est pas source de df vers une partie de la clé).

ex : soit la relation Rémunération(grade,ancienneté,salaire)
grade,ancienneté $\rightarrow$ salaire (le salaire est fonction du grade et de l'ancienneté)
si les plages de salaire ne se chevauchent pas on a également :
salaire $\rightarrow$ grade
La relation Rémunération n'est pas en FNBC.

Exemple 4.1 (Exemple 1) *(P. Mathelot et al. "Merise, exercices corrigés". Les Editions d'organisation)*
Soit R(A,B,C) la relation telle que $A \rightarrow B$, $B \rightarrow C$. Quelle est la clé de R et sa forme normale?
Réponse : la clé est A (car $A \rightarrow B$ et $A \rightarrow C$), elle est en 2FN (car $A \rightarrow C$ est transitive donc pas 3FN)
La relation suivante est-elle conforme aux df de R :

A	*B*	*C*
a1	*b1*	*c1*
a1	*b1*	*c2*
a3	*b2*	*c1*
a4	*b3*	*c3*

$A \rightarrow B$ OK, mais $B \rightarrow C$ non car (b1,c1) et (b1,c2) appartiennent à $\pi_{B,C}(R)$.

Exemple 4.2 (Exemple 2) *(P. Mathelot et al. "Merise, exercices corrigés". Les Editions d'organisation)*
Pour constituer une BD, on dispose des éléments suivants : no-etudiant, no-uv, nom-uv, nbre-heures-enseignement, no-ss-enseignant, nom-enseignant, grade, indice, ufr-rattachement, nbre-enseignants-ufr, resultat-uv, nom-etu
On fait les hypothèses suivantes :

- *un étudiant peut s'inscrire à plusieurs uv*

– *resultat-uv caractérise un étudiant pour une uv donnée*
– *nbre-heures-enseignements est spécifique à une uv*
– *chaque uv n'est assurée que par un enseignant*
– *un enseignant est rattaché à un seul ufr*
– *à chaque grade correspond son indice*

questions : trouver les df (faire un graphe), donner les relations et les clefs pour obtenir un schéma en BCNF

4.3 contraintes en SQL

Lorsque l'on crée une table en SQL on peut ajouter des contraintes sur les colonnes :
Les Contraintes de colonne :
- PRIMARY KEY (clé primaire) : dans ce cas Oracle crée un index sur cette colonne
- UNIQUE : impose qu'il n'y ait pas de redondance sur cette colonne
- NOT NULL : impose que cette colonne ait toujours une valeur
- CHECK : `CHECK (nom_colonne IN (val1, val2, ...))]` impose que la valeur de la colonne appartienne à la liste donnée
- FOREIGN KEY (clé étrangère) : `FOREIGN KEY (nom_colonne) REFERENCES nom_table [(nom_colonne)]` vérifie que la valeur de la colonne apparait dans l'autre table.

Il est possible de mettre plusieurs contraintes pour une même colonne. Une contrainte a toujours un nom défini par l'utilisateur ou par le système. Une contrainte est nommées grâce au mot clef `CONSTRAINT` suivi du nom que l'on désire lui donner, puis du corps de la contrainte. Les contraintes de table portent sur plusieurs colonnes, elles sont définies en fin de table.
ex:

```
CREATE TABLE Personnel
(cdprs CHAR(3) PRIMARY KEY,
nom VARCHAR(25) NOT NULL,
prnm VARCHAR(10),
datnais DATE CONSTRAINT dat_attendue NOT NULL,
cdsrv CHAR(5) FOREIGN KEY cdsrv REFERENCES service(cdsrv))
```

A l'insertion les contraintes sont vérifiées

```
INSERT INTO Personnel VALUES('023','Dupont',NULL,NULL,'07');
>erreur la contraite dat_attendue est violée.
```

5 Transactions concurrentes

Définition 5.1 *Une transaction est une séquence continue d'actions(lecture/écriture) sur la base de données.*

Propriétés des transactions :

- atomicité : tout ou rien
- cohérence : Chaque transaction doit laisser la BD dans un état cohérent après l'avoir prise dans un état cohérent. (Le SGBD vérifie les contraintes d'intégrité avant chaque écriture).
- isolation : les mises à jour faites par une transaction ne sont visibles à l'extérieur de celle-ci qu'après validation
- durabilité : après la fin de la transaction les mises à jour sont définitives même en cas de panne ultérieur

Problèmes liés aux accès concurrents :

Perte de mise à jour :

utilisateur 1	utilisateur 2
lire A ->a	
-	lire A ->b
-	b++
-	ecrire b ->A
a <- a*2	-
écrire a->A	-

La mise à jour de l'utilisateur 2 est perdue.

Incohérence : violation de contrainte d'intégrité . CI: A=B

utilisateur 1	utilisateur 2
A*2->A	-
-	A+1 ->A
-	B+1->B
-	-
-	-
B*2->B	-

CI est violée!

Lecture impropre :

utilisateur1	utilisateur2
lire(x)	-
x:=x-n	-
écrire(x)	-
-	lire(x)
-	x:=x+m
-	-
lire(y)	
abort	

La valeur de x lue par l'utilisateur2 n'est plus valide.

5.1 mécanismes pour assurer la concurrence

Deux approches :

- approche pessimiste : verrouillage (on fait en sorte que l'on n'ait pas d'exécution non correcte)

– approche optimiste : on laisse s'exécuter les transactions sans contraintes et au moment de la validation s'il y a un conflit on redéfait toute la transaction

La seule solution implantée de façon industrielle est le verrouillage.

Définition 5.2 (verrou) *Un verrou est une variable d'état associée à un objet X de la base indiquant son état vis-à-vis des opérations de lecture/écriture*

Verrou ternaire : verrouillé en lecture, verrouillé en écriture ou libre. Si l'objet est libre alors il peut être écrit et devient verrouillé en écriture ou bien lu il devient alors verrouillé en lecture. Si l'objet est verrouillé en lecture alors il peut être lu mais pas écrit. Si l'objet est verrouillé en écriture alors il ne peut être ni lu ni écrit.

Définition 5.3 (verrouillage deux phases) *C'est un protocole de verrouillage qui garantit la serialisabilité mais n'empèche pas l'interblocage.Avant de lire/écrire un élément chaque transaction demande à émettre un verrou. Les verrous sont relachés en fin de transaction.*

utilisateur 1	utilisateur 2
en lecture sur A	en lecture sur A
veut modifier A	en lecture en A
ATTENTE	-
-	VALIDATION
modifie A	-

Si deux transactions s'attendent =>interblocage (Dead lock),

utilisateur 1	utilisateur 2
en lecture sur A	en lecture sur B
veut modifier B	-
ATTENTE	veut modifier A
-	ATTENTE ->DEAD LOCK
-	ABORT
peut modifier B	

Pour éviter l'interblocage :

– priorités entre transactions (pessimiste : une transaction de faible priorité devra toujours attendre celles de plus forte priorité et jamais l'inverse)
– détection des interblocage et assasinat d'une des transactions responsable (la moins longue, la plus récente,...)

5.2 reprise

echec d'une transaction entraîne un état incohérent de la BD.

Définition 5.4 (mise à jour différée) *Ne faire les écritures qu'au moment de la validation de la transaction. Les modifications sont enregistrées dans le journal lors de la validation. La reprise consiste à parcourir le journal pour refaire les opérations d'écriture* *des transactions validées depuis le dernier point de reprise. Les transactions non validées n'ont rien modifié donc il n'y a rien à faire pour elles.*

Définition 5.5 (mise à jour immédiate) *Les écritures sont effectuées immédiatement. Les anciennes valeurs des informations modifiées sont enregistrées dans le journal. La reprise consiste à refaire les opérations des transactions validées depuis le point de reprise et défaire toutes les opérations des transactions invalidées.*

Sous Oracle :

- pour valider une transaction et libérer les verrous : COMMIT
- pour annuler une transaction et libérer les verrous : ABORT
- pour revenir à l'état d'avant la transaction en cours : ROLLBACK (revient au dernier COMMIT ou au dernier jalon ou au jalon précisé TO SAVEPOINT)
- pour poser des jalons SAVEPOINT nomdupointdesauvegarde

5.3 les droits

Le propriétaire d'une table peut gérer les droits d'accès à sa table sous oracle : instruction GRANT (cf manuel)

6 Optimisation

6.1 Optimisation physique

Le but est de modifier le stockage des données afin que les accès soient plus rapides.

6.1.1 Les index

Objectif : accélérer la sélection ou le tri dans une table (seulement quand on s'intéresse aux valeurs de la clé)

Définition 6.1 *Un index est une sorte de table des matières qui donne pour chaque valeur d'une clé, l'adresse de l'article correspondant à cette clé.*

Un index augmente les performances de recherche d'élément mais diminuent les performances de mise à jour.
Il faut mettre des index sur des éléments qui sont plus lus qu'écrits. Créer la table, insérer les lignes puis mettre un index sinon c'est plus long.
Sous Oracle : On peut mettre des index explicitement ou en utilisant la contrainte PRIMARY KEY.

```
CREATE INDEX <nom> ON <nomtable>(<col1,col2,...>);
```

exemple :

```
CREATE TABLE Lespays (nom VARCHAR(30), codecapitale CHAR(3), population NUMBER);

CREATE INDEX i_pays ON Lespays(nom,capitale);
```

Cela signifie qu'une table des matières est associée à la table Lespays avec pour index le nom et le codecapitale. Pour un nom et une capitale donnée on a l'adresse de l'enregistrement du pays correspondant.

```
SELECT population FROM Lespays WHERE NOM like 'A%'    => optimisee

SELECT population FROM Lespays
WHERE codecapitale='PRS'                => pas optimisee
AND NOM like 'A%'                       => optimisee
```

Si une table comporte plusieurs index, l'index correspondant à la clef primaire est appelé index primaire et les autres sont des index secondaires.

6.1.2 Les clusters

Définition 6.2 *Un cluster est un stockage groupé des lignes de plusieurs tables ayant une caractéristique commune. Ce qui permet d'accélérer les accès par navigation ou par jointure.*

On peut stocker ensemble des objets liés par une association.
Sous Oracle : Donner un nom pour le cluster qui représente l'association à grouper, préciser le nom des colonnes sur lesquelles se fera le regroupement puis définir les tables en précisant qu'elles appartiennent au cluster. Enfin créer un index pour le cluster.
ex :

```
CREATE TABLE LesVilles
(numville CHAR(5), numpays CHAR(5), nom VARCHAR(20), pop NUMBER, nbhot NUMBER)
```

Si l'on veut toutes les villes d'un pays qui commence par une certaine lettre :

```
SELECT v.nom
FROM Lesvilles v, Lespays p
WHERE v.numpays=p.codepays
AND p.nom like 'X%'                      => requete non optimisée
```

Il faut créer un cluster qui pour un codepays donnée regroupe les informations sur le pays avec les infos sur toutes les villes de ce pays.

```
CREATE CLUSTER paysvilles(codepays CHAR(5));

CREATE TABLE LesPays (numpays CHAR(5), nom VARCHAR(20), pop NUMBER)
CLUSTER paysvilles (numpays);
```

```
CREATE TABLE LesVilles
(numville CHAR(5), numpays CHAR(5), nom VARCHAR(20), pop NUMBER, nbhot NUMBER)
CLUSTER paysvilles (numpays);

CREATE INDEX i_pays ON CLUSTER paysvilles; --l'index est obligatoire
```

stockage :

p1	France	57000000	v1	Paris	...
			v2	Toulouse	...
			v3	Angers	...
p2	Italie	.....			

6.2 Optimisation logique

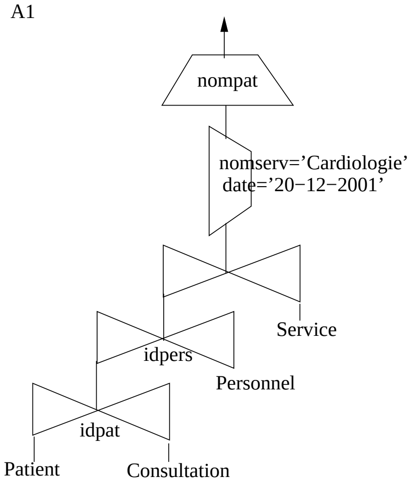
Transformation des requêtes par le biais de l'algèbre relationnelle
Les relations unaires (π et σ) réduisent la taille des relations. Les relations binaires ($\bowtie$, $\times$, $\cup$) accroissent la taille des résultats
Optimisation : descendre les relations unaires (règles de réécriture)

- $\cap$, $\cup$, $\setminus$ règles de la théorie des ensembles
- règles spécifiques :
  - $R \bowtie S = S \bowtie R$: vrai à l'ordre des colonnes près
  - $R \bowtie_{A=B} (S \bowtie_{C=D} T) = (R \bowtie_{A=B} S) \bowtie_{C=D} T$: vrai si B et C sont des attributs de S et A de R et D de T, si B $\notin$ T alors $R \bowtie_{A=B} T$ est impossible.
  - descente de restriction
    - $\sigma_c(R \bowtie S) = \sigma_c(R) \bowtie S$: vrai si c est une condition sur R
    - $\sigma_{cetd}(R) = \sigma_c(\sigma_d(R)) = \sigma_c(R) \cap \sigma_d(R)$
    - $\sigma_{coud}(R) = \sigma_c(R) \cup \sigma_d(R)$
    - $\sigma_{\not c}(R) = R \setminus \sigma_c(R)$
  - descente de projection
    - $\pi_{A,B,C}(R \cup S) = \pi_{A,B,C}(R) \cup \pi_{A,B,C}(S)$
    - $\pi_A(R \cap S) \neq \pi_A(R) \cap \pi_A(S)$
    - $\pi_A(R \setminus S) \neq pi_A(R) \setminus \pi_A(S)$
    - $\pi_L(R \bowtie_{A=B} S) = \pi_L(\pi_M(R) \bowtie_{A=B} \pi_N(S)$: M attributs de L$\in$ R suivis de l'attribut A, N attributs de L $\in$ S suivis de D.
      ex : $\pi_{p.nom,p.pop,v.nbh}(Lesvilles \bowtie_{numpays=codepays} LesPays)$
      $\pi_{p.nom,p.pop,v.nbh}(\pi_{nbh,numpays}(LesVilles) \bowtie_{numpays=codepays} \pi_{p.nom,p.pop,codepays} LesPays)$

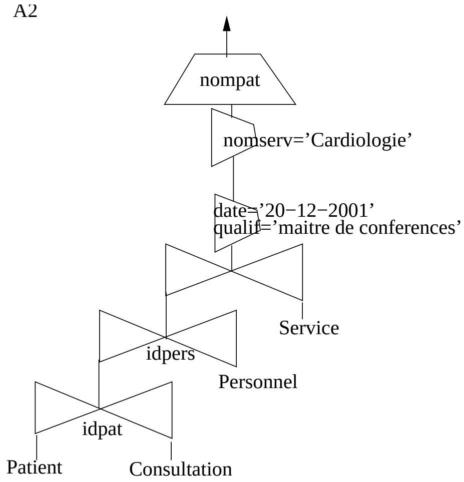
ex : Le nom des patients qui ont eu une consultation dans le service "Cardiologie" le 20-12-2001 et de profession "Maître de conférences"
Patient(idpat,...)
Consultation(idpers,idpat,date)

Personnel(idpers,...)
Service(codeserv,...)

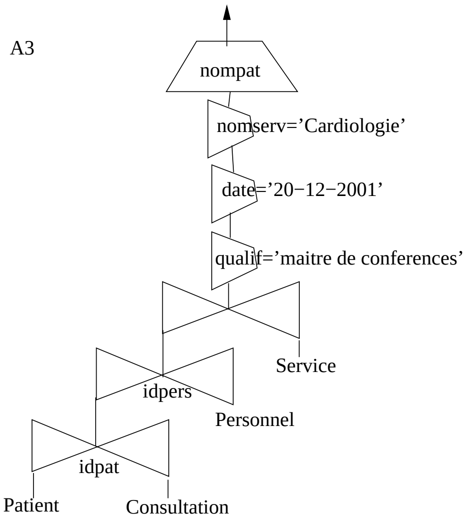
A1



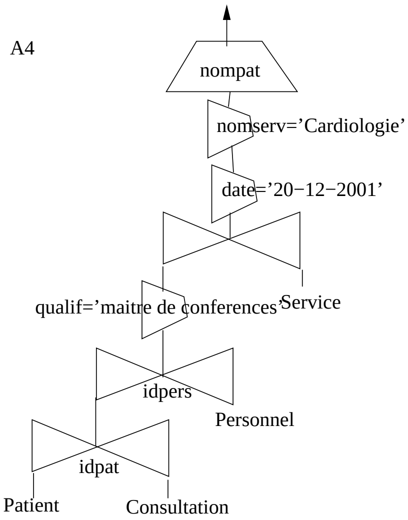
A2



A3



A4



A1 pas optimisé, il faut d'abord descendre les restrictions. A2=σ_{et} (A1). A3=σ_{et} (A2). A4= descente de σ_{qualif} dans A3. A5= descente de σ_{qualif} dans A4. A6= descente de σ_{qualif} dans A5. A7= descente de σ_{date} dans A6. A8= descente de σ_{date} dans A7. A9= descente de σ_{date} dans A8. A10= descente de $\sigma_{nomserv}$ dans A9.