Conservatoire National des Arts et Métiers

292 Rue St Martin 75141 Paris Cedex 03

Exercices Dirigés en Bases de Données

M. Scholl, B. Amann, P. Rigaux et D. Vodislav

Solutions

27 septembre 2001

Table des matières

Chapitre 1

Conception

1.1 Interprétation de schémas entité/association

1.1.1 Centre médical

On vous donne un schémas E/A (figure 1.1) représentant des visites dans un centre médical. Répondez aux questions suivantes **en fonction des caractéristiques de ce schéma** (i.e.: indiquez si la situation décrite est représentable, indépendamment de sa vraisemblance).

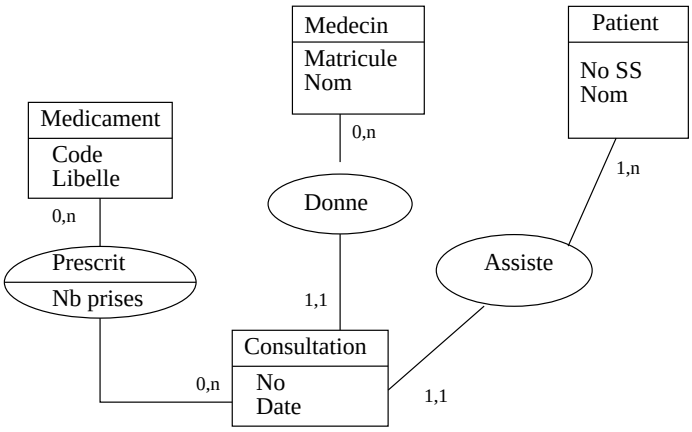


FIG. 1.1 – *Centre médical*

Exercice A : Un patient peut-il effectuer plusieurs visites ?
Exercice B : Un médecin peut-il recevoir plusieurs patients dans la même consultation ?
Exercice C : Peut-on prescrire plusieurs médicaments dans une même consultation ?
Exercice D : Deux médecins différents peuvent-ils prescrire le même médicament ?

Solution :

1. *Bien sûr.*

2. *Non (un patient par consultation).*
3. *Oui.*
4. *Oui (pas de rapport entre un médecin et une consultation).*

1.1.2 Tournoi de tennis

Le second schéma (figure 1.2) représente des rencontres dans un tournoi de tennis.

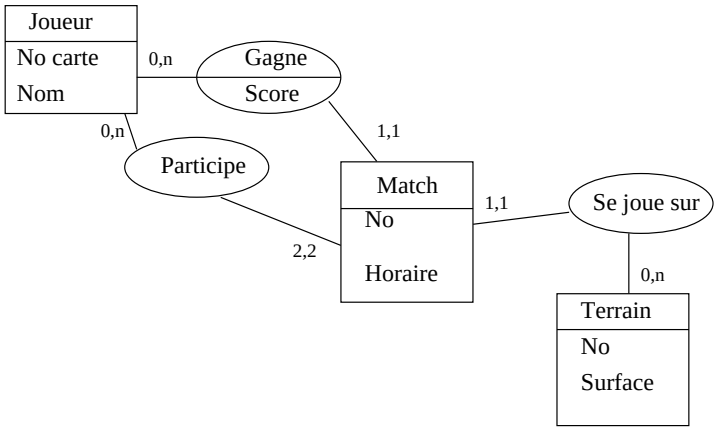


FIG. 1.2 – *Tournoi de tennis*

Exercice A : Peut-on jouer des matchs de double ?
Exercice B : Un joueur peut-il gagner un match sans y a voir participé ?
Exercice C : Peut-il y avoir deux matchs sur le même terrain à la même heure ?

1.1.3 Un journal

Pour vous entraîner : voici le schéma E/A (figure 1.3 du système d'information (très simplifié) d'un quotidien.

Exercice A : Un article peut-il être rédigé par plusieurs journalistes ?
Exercice B : Un article peut-il être publié plusieurs fois dans le même numéro ?
Exercice C : Peut-il y avoir plusieurs articles sur le même sujet dans le même numéro ?
Exercice D : ...

1.2 Modèle réseau

Exercice A : Pour chacun des schémas E/A donnés précédemment, construire le schéma réseau correspondant (types d'enregistrements et types de sets).
Exercice B : Exprimez les requêtes suivantes :

1. La liste des consultations du Dr Duchemin.
2. Les patients qui ont consulté le 15 mai 1997.

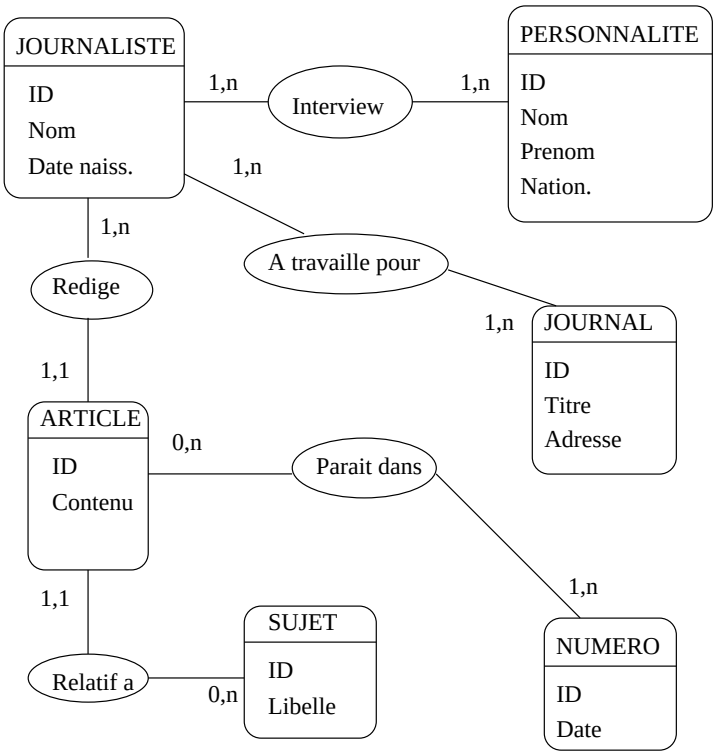


FIG. 1.3 – *Journal*

3. A quelles dates a-t-on prescrit de l'aspirine ?
4. Les joueurs qui ont gagné au moins un match.
5. Les joeurs qui se sont rencontrés dans un match.

1.3 Modèle relationnel

Exercice A : Pour chacun des schémas E/A donnés précédemment, construire le schéma relationnel correspondant. Indiquez précisément :

- La clé primaire.
- Les clés étrangères.
- Les contraintes éventuelles.

Solution :

Exemple pour le centre médical :

- *Médicament (***Code**, *Libellé).*
- *Consultation (***ID-consultation**, *Matricule, NO-SS, Date). Matricule et NO-SS sont les clés étrangères.*
- *Prescription (***Code-médicament, ID-consultation**, *Nb-prises)*
- *Médecin (***Matricule**, *Nom).*
- *Patient (***NO-SS**, *Nom).*

Exercice B : Donnez la commande **Create Table** pour les tables $Consultation$ et $Match$.

Solution :

Exemple pour la table Consultation :

Create Table *Consultation*
(Id-consultation **NUMBER(10)**,
Matriculeé **NUMBER(10)**,
NO-SS **NUMBER(10)**,
Date-consultation **DATE**,
PRIMARY KEY *(Id-consultation)*,
FOREIGN KEY *Matricule* **REFERENCES** *Médecin*,
FOREIGN KEY *NO-SS* **REFERENCES** *Patient)*

Exercice C : Répondez aux mêmes requêtes que celles données pour le modèle réseau (en algèbre et en SQL).

1.4 Rétro-conception

On trouve dans un SGBD relationnel les relations ci-dessous. Les clés primaires sont soulignées, mais pas les clés étrangères.

IMMEUBLE (**Adresse**, Nb-étages, Date-construction, Nom-Gérant)
APPART (**Adresse, Numéro**, Type, Superficie, Etage)
PERSONNE (**Nom**, Age, Code-Profession)
OCCUPANT (**Adresse, Numéro-Appart, Nom-Occupant**, Date-arrivée, Date-départ)
PROPRIÉTÉ (**Adresse, Nom-Propriétaire**, Quote-part)
TYPE-APPART (**Code**, Libellé)
PROFESSION (**Code**, Libellé)

Exercice A : Identifier les clés étrangères dans chaque relation.
Exercice B : Reconstruire le schéma E/A.
Exercice C : Existe-t-il des contraintes d'intégrité ? Lesquelles ?
Exercice D : Certaines données du schéma relationnel résultent-elles d'optimisation ?

Chapitre 2

Algèbre Relationelle

2.1 Sélection et Projection

Soit la relation

PERSONNE		
Nom	Age	Ville
Marc	29	Paris
Catherine	32	Lyon
Sophie	54	Paris
Claude	13	Montpellier
Serge	40	Lyon

Exercice A : Donnez les résultats des requêtes suivantes :

Requête 1 : $\sigma_{Age=30}(PERSONNE)$ (sélection)

Requête 2 : $\pi_{Age}(PERSONNE)$ (projection)

Requête 3 : $\pi_{Age}(\sigma_{Nom='Serge'}(PERSONNE))$ (projection, sélection)

Exercice B : Exprimez les requêtes suivantes en algèbre relationnelle:

Requête 1 : les personnes (nom, âge, ville) qui habitent Paris.

Solution :
$\sigma_{Ville='Paris'}(PERSONNE)$

Requête 2 : les personnes (nom, âge, ville) qui ont moins de 30 ans.

Solution :
$\sigma_{Age<30}(PERSONNE)$

Requête 3 : les villes dans la relation PERSONNE.

Solution :
$\pi_{Ville}(PERSONNE)$

Requête 4 : les noms des personnes habitant à Paris.

Solution :
$\pi_{Nom}(\sigma_{Ville='Paris'}(PERSONNE))$

2.2 Jointure relationnelle

Exercice A : Soient **R** et **S** les relations

R	
A	B
a	b
a	f
c	b
d	e

S	
B	C
b	c
e	a
b	d
g	b

où les attributs A, B, C sont définis sur les domaines des lettres de l'alphabet.
Donnez le résultat des requêtes suivantes :

Requête 1 : $R \bowtie S$ (jointure naturelle).

Requête 2 : $\sigma_{A=C}(\rho_{B/B'}(R) \times S)$ (équi-jointure).

Requête 3 : $R \rhd\!\!< S = \pi_R(R \bowtie S)$ (semijoin).

Solution :

$R \bowtie S$		
A	*B*	*C*
a	*b*	*c*
a	*b*	*d*
c	*b*	*c*
c	*b*	*d*
d	*e*	*a*

$\sigma_{A=C}(\rho_{B/B'}(R) \times S)$			
A	*B'*	*B*	*C*
a	*b*	*e*	*a*
a	*f*	*e*	*a*
c	*b*	*b*	*c*
d	*e*	*b*	*d*

$R \rhd\!\!< S$	
A	*B*
a	*b*
c	*b*
d	*e*

Exercice B : Est-ce que les équations suivantes sont vraies ?

$$\pi_{A,B}(R \bowtie S) = R \tag{2.1}$$
$$\pi_{B,C}(R \bowtie S) = S \tag{2.2}$$

Solution :
NON:

$\pi_{A,B}(R \bowtie S)$	
A	*B*
a	*b*
c	*b*
d	*e*

$\pi_{B,C}(R \bowtie S)$	
B	*C*
b	*c*
b	*d*
e	*a*

2.3 Auto-Jointure et Renommage

Soit **T**(A,B) une relation où A et B prennent leurs valeurs dans le même domaine. Supposons qu'on veuille sélectionner les seuls n-uplets $< a,b >$ tels que $< b,a >$ est également un n-uplet de **T**.

Exprimez cette opération par une expression de l'algèbre relationnelle.

Solution :

1. *solution :*
   (a) *On fait une copie de* **T** *dans* **S***(A,B)* $S := T$
   (b) *On* renomme *l'attribut A en* A_1 *et B en* B_1 $S := \rho_{A/A_1, B/B_1}(S)$
   (c) **S** *a maintenant pour schéma* $\mathbf{S}(A_1, B_1)$
   (d) *Le résultat est* $T \rhd\!\lhd_{B=A_1 \wedge A=B_1} S$
2. *solution :*

$$T \cap \rho_{B/A, A/B}(T)$$

Chapitre 3

Algèbre : Compagnie d'Assurance

3.1 Schéma

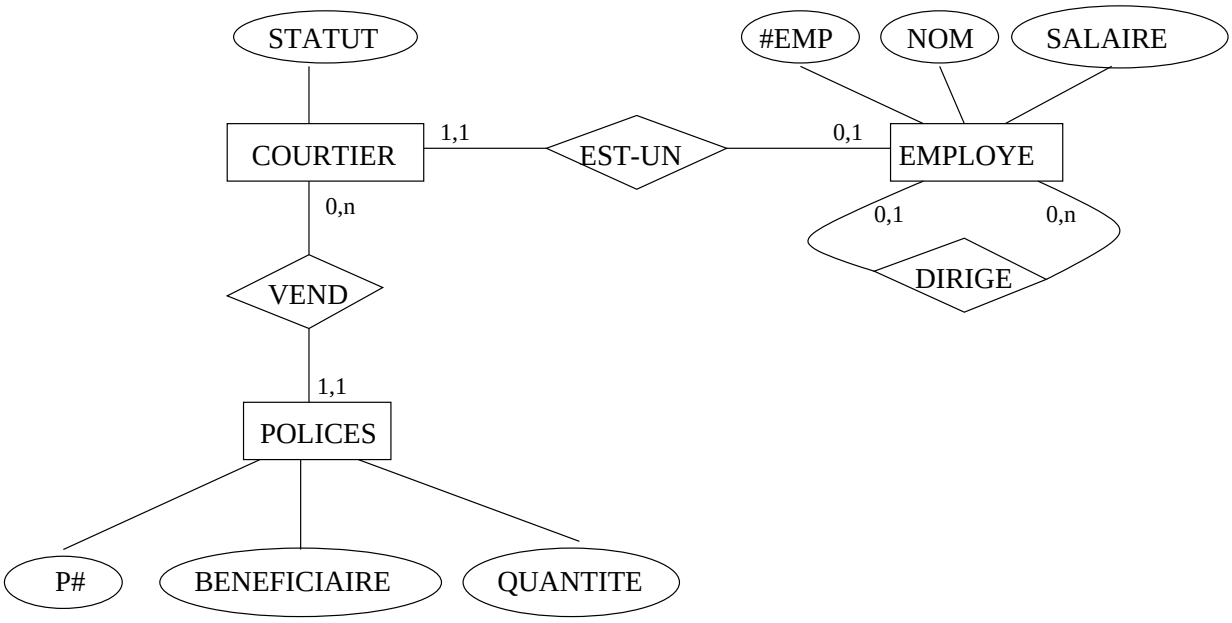


FIG. 3.1 – *Compagnie d'assurances*

Exercice A : décrire ce schéma Entité-Relation (en français)
Exercice B : représenter le diagramme Entité-Relation précédent dans le modèle relationnel.

Solution :
(les attributs clés sont soulignés)
POLICES*(***P#***, BENEFICIAIRE, QUANTITE, #EMP)*
EMPLOYE*(***#EMP***, NOM, SALAIRE, #EMP_D)*
COURTIER*(***#EMP***, STATUT)*

3.2 Requêtes

Exercice A : exprimer les requêtes suivantes en algèbre relationelle:

Requête 1 : les numéros des polices vendues à plus de 20 exemplaires.

Solution :
$\pi_{\#P}(\sigma_{QUANTITE>20}(POLICES))$

Requête 2 : les noms des employés qui gagnent moins que 6000 francs.

Solution :
$\pi_{NOM}(\sigma_{SALAIRE<6000}(EMPLOYE))$

Requête 3 : les noms de tous les courtiers.

Solution :
$\pi_{NOM}(COURTIER \bowtie EMPLOYE)$

Requête 4 : les bénéficiaires d'au moins 21 polices avec le même numéro (> 20).

Solution :
$\pi_{BENEFICIAIRE}(\sigma_{QUANTITE>20}(POLICES))$

Requête 5 : les noms de courtiers dirigés par l'employé 113.

Solution :

1. *solution:* $\pi_{NOM}(\sigma_{\#EMP\_D=113}(COURTIER \bowtie EMPLOYE))$
2. *solution (plus efficace):* $\pi_{NOM}(COURTIER \bowtie \sigma_{\#EMP\_D=113}(EMPLOYE))$

Requête 6 : les salaires des courtiers stagiaires (STATUT='Stagiaire').

Solution :
$\pi_{SALAIRE}(\sigma_{STATUT='Stagiaire'}COURTIER \bowtie EMPLOYE)$

Chapitre 4

Algèbre - SQL: Employés - Départements

4.1 Schéma

Les exemples suivants sont tirés des sources de la société Oracle.

4.1.1 Relation des Employés (EMP)

EMP(**ENO**, ENOM, PROF, DATEEMB, SAL, COMM, DNO)

ENO : numéro d'employé, clé
ENOM : nom de l'employé
PROF : profession (directeur n'est pas une profession)
DATEEMB : date d'embauche
SAL : salaire
COMM : commission (un employé peut ne pas avoir de commission)
DNO : numéro de département auquel appartient l'employé

4.1.2 Relation des Départements (DEPT)

DEPT(**DNO**, DNOM, DIR, VILLE)

DNO : numéro de département, clé
DNOM : nom du département
DIR : directeur du département
VILLE : lieu du département (ville)

4.2 Opérations Algébriques

Soit l'exemple suivant :

EMP

ENO	ENOM	PROF	DATEEMB	SAL	COMM	DNO
10	Joe	Ingénieur	1.10.93	4000	3000	3
20	Jack	Technicien	1.5.88	3000	2000	2
30	Jim	Vendeur	1.3.80	5000	5000	1
40	Lucy	Ingénieur	1.3.80	5000	5000	3

DEPT

DNO	DNOM	DIR	VILLE
1	Commercial	30	New York
2	Production	20	Houston
3	Développement	40	Boston

Exercice A : Calculer $\sigma_{sal<5000}(EMP)$.

Solution :

ENO	*ENOM*	*PROF*	*DATEEMB*	*SAL*	*COMM*	*DNO*
10	*Joe*	*Ingénieur*	*1.10.93*	*4000*	*3000*	*3*
20	*Jack*	*Technicien*	*1.5.88*	*3000*	*2000*	*2*

Exercice B : Calculer **EMPbis** = $\rho_{ENO/ENO'}(\pi_{ENO,COMM}(EMP))$

Solution :

EMPbis

ENO'	*COMM*
10	*3000*
20	*2000*
30	*5000*
40	*5000*

Exercice C : Calculer $\pi_{ENO,SAL}(EMP) \bowtie_{SAL=COMM} (EMPbis)$

Solution :

ENO	*SAL*	*ENO'*	*COMM'*
20	*3000*	*10*	*3000*
30	*5000*	*30*	*5000*
40	*5000*	*40*	*5000*
30	*5000*	*40*	*5000*
40	*5000*	*30*	*5000*

Exercice D : Exprimer par une phrase ce qu'on obtient en évaluant les requêtes précédentes.

Solution :

- *Expression 1: on obtient les employés dont le salaire est inférieur à 5000.*
- *Expression 2: on obtient le numéro et la commission des employés.*
- *Expression 3: on obtient les couples de numéros d'employés dont le premier a un salaire qui est égal à la commission du deuxième.*

Exercice E : Quelle est l'expression de l'algèbre relationnelle qui permettrait d'obtenir le nom et la profession de l'employé de numéro 10.

Solution :

$$\pi_{ENOM,PROF}(\sigma_{ENO=10}(EMP))$$

Exercice F : Idem pour la liste des noms des employés qui travaillent à New York.

Solution :

$$\pi_{ENOM}(EMP \bowtie (\sigma_{VILLE='NewYork'}(DEPT)))$$

Exercice G : Idem pour avoir le nom du directeur du département "Commercial".

Solution :

$$\pi_{ENOM}(EMP \bowtie_{ENO=DIR} \pi_{DIR}(\sigma_{DNOM='Commercial'}(DEPT)))$$

4.3 Requêtes

- Exprimer les requêtes Q1 à Q18 à l'aide de l'algèbre relationnelle.
- Exprimer en SQL les requêtes Q1 à Q24.

4.3.1 Interrogation d'une seule Relation

Requête 1 : Donner tous les n-uplets de DEPT.

Solution :
Algèbre : $DEPT$
SQL :

```
SELECT * FROM DEPT;
```

Requête 2 : Donner tous les n-uplets de EMP.

Solution :
Algèbre : EMP
SQL :

```
SELECT * FROM EMP;
```

Requête 3 : Donner les noms et les salaires des employés.

Solution :
Algèbre : $\pi_{ENOM,SAL}(EMP)$
SQL :

```
SELECT ENOM, SAL
  FROM EMP;
```

Requête 4 : Donner les professions des employés (après élimination des duplicats).

Solution :
Algèbre : $\pi_{PROF}(EMP)$
SQL :

```
SELECT DISTINCT PROF
  FROM EMP;
```

Requête 5 : Donner les dates d'embauche des techniciens.

Solution :
Algèbre : $\pi_{DATEEMB}(\sigma_{PROF='TECHNICIEN'}(EMP))$
SQL :

```
SELECT DATEEMB
  FROM EMP
 WHERE PROF='TECHNICIEN';
```

4.3.2 Jointures

Requête 6 : Faire le produit cartésien entre EMP et DEPT.

Solution :
Algèbre : $EMP \times DEPT$
SQL :

```
SELECT *
  FROM EMP, DEPT;
```

Requête 7 : Donner les noms des employés et les noms de leur département.

Solution :
Algèbre : $\pi_{ENOM,DNOM}(EMP \bowtie DEPT)$
SQL :

```
SELECT ENOM, DNOM
  FROM EMP, DEPT
 WHERE EMP.DNO=DEPT.DNO;
```

Requête 8 : Donner les numéros des employés travaillant à BOSTON.

Solution :
Algèbre : $\pi_{ENO}(EMP \bowtie \sigma_{VILLE='BOSTON'}(DEPT))$
SQL :

```
SELECT ENO
  FROM EMP, DEPT
 WHERE EMP.DNO=DEPT.DNO AND VILLE='BOSTON';
```

Requête 9 : Donner les noms des directeurs des départements 1 et 3. Attention : directeur n'est pas une profession!

Solution :
Algèbre : $\pi_{ENOM}(\sigma_{DNO=1 \vee DNO=3}(DEPT) \bowtie_{DIR=ENO} EMP)$
SQL :

```
SELECT ENOM
  FROM EMP, DEPT
 WHERE (DEPT.DNO=1 OR DEPT.DNO=3) AND DIR = ENO;
```

ou

```
SELECT ENOM
  FROM EMP, DEPT
 WHERE DEPT.DNO IN (1,3) AND DIR = ENO;
```

Requête 10 : Donner les noms des employés travaillant dans un département avec au moins un ingénieur.

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \pi_{DNO}(\sigma_{PROF='INGENIEUR'}(EMP)) \\ R2 &:= \pi_{ENOM}(EMP \bowtie R1) \end{aligned}$$

SQL :

```
SELECT E2.ENOM
  FROM EMP E1, EMP E2
 WHERE E1.DNO = E2.DNO
       AND E1.PROF = 'INGÉNIEUR';
```

Requête 11 : Donner le salaire et le nom des employés gagnant plus qu'un (au moins un) ingénieur.

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \rho_{SAL/SAL1}(\pi_{SAL}(\sigma_{PROF='INGENIEUR'}(EMP))) \\ R2 &:= \pi_{ENOM,SAL}(EMP \bowtie_{SAL>SAL1} R1) \end{aligned}$$

SQL :

```
SELECT E1.ENOM, E1.SAL
  FROM EMP E1, EMP E2
 WHERE E2.PROF='INGENIEUR'
       AND E1.SAL > E2.SAL;
```

ou

```
SELECT ENOM, SAL FROM EMP
 WHERE SAL > ANY (SELECT SAL
                  FROM EMP
                  WHERE PROF='INGENIEUR');
```

Requête 12 : Donner le salaire et le nom des employés gagnant plus que **tous les ingénieurs**.

Solution :
SQL :

```
SELECT ENOM, SAL FROM EMP
       WHERE SAL > ALL (SELECT SAL
                        FROM EMP
                        WHERE PROF='INGENIEUR');
```

Requête 13 : Donner les noms des employés et les noms de leurs directeurs.

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \rho_{ENOM/DIRNOM}(\pi_{ENOM,DNO}(EMP \bowtie_{ENO=DIR} DEPT)) \\ R2 &:= \pi_{ENOM,DNO}EMP \\ R3 &:= \pi_{ENOM,DIRNOM}(R1 \bowtie R2) \end{aligned}$$

- *R1(DIRNOM,DNO) : les départements avec les noms de leur directeur*
- *R2(ENOM,DNO) : les employés avec les numéros de leur département*

– *R3(ENOM,DIRNOM) : les employés (ENOM) avec les noms de leur directeur*

SQL :

```
SELECT E1.ENOM, E2.ENOM
  FROM EMP E1, EMP E2, DEPT D
 WHERE E1.DNO=D.DNO AND E2.ENO = D.DIR;
```

Requête 14 : Trouver les noms des employés ayant le même directeur que JIM. Attention : un employé peut être directeur de plusieurs départements.

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \pi_{DIR}(\sigma_{ENOM='JIM'}(EMP) \bowtie DEPT) \\ R2 &:= DEPT \bowtie R1 \\ R3 &:= \pi_{ENOM}(\sigma_{ENOM<>'JIM'}(EMP) \bowtie R2) \end{aligned}$$

– *R1(DIR) : le numéro du directeur de JIM*
– *R2(...) : les départements avec le même directeur*
– *R3(ENOM) : les noms des employés ayant le même directeur que JIM.*

SQL :

```
SELECT ENOM
  FROM EMP
 WHERE ENOM <> 'JIM'
       AND DNO IN (SELECT D2.DNO
                     FROM EMP,
                          DEPT D1, DEPT D2
                    WHERE ENOM='JIM'
                          AND D1.DNO = EMP.DNO
                          AND D1.DIR = D2.DIR);
```

Requête 15 : Donner le nom et la date d'embauche des employés embauchés avant leur directeur; donner également le nom et la date d'embauche de leur directeur.

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \pi_{DNO,ENOM,DATEEMB}(EMP \bowtie_{DIR=ENO} DEPT) \\ R2 &:= \rho_{ENOM/DIRNOM,DATEEMB/DIRDATE}(R1) \\ R3 &:= \pi_{ENOM,DATEEMB,DIRNOM,DIRDATE}(\sigma_{DIRDATE<DATEEMB}(EMP \bowtie R2)) \end{aligned}$$

– *R1(DNO,ENOM,DATEEMB) : le nom et la date d'embauche du directeur du dept. DNO.*
– *R2(DNO,DIRNOM,DIRDATE) : renommage des attributs*
– *R3(ENOM,DATEEMB,DIRNOM,DIRDATE) : résultat*

SQL :

```
SELECT E1.ENOM, E1.DATEEMB, E2.ENOM, E2.DATEEMB
  FROM EMP E1, EMP E2, DEPT D
 WHERE E2.ENO=D.DIR
       AND E1.DNO=D.DNO
       AND E1.DATEEMB<E2.DATEEMB;
```

Requête 16 : Donner les départements qui n'ont pas d'employés.

Solution :
Algèbre : $DEPT - (DEPT \rhd\!\!< EMP)$

SQL :

```
SELECT *
  FROM DEPT
 WHERE DNO NOT IN (SELECT DNO FROM EMP);
```

Requête 17 : Donner les noms des employés du département COMMERCIAL embauchés le même jour qu'un employé du département PRODUCTION.

Solution :
Algèbre :

$$R1 := \pi_{DATEEMB}(EMP \bowtie \sigma_{DNOM='PRODUCTION'}(DEPT))$$
$$R2 := \pi_{ENOM}((EMP \bowtie \sigma_{DNOM='COMMERCIAL'}\ DEPT) \bowtie R1)$$

SQL :

```
SELECT DISTINCT ENOM
  FROM EMP E1, DEPT D1, EMP E2, DEPT D2
 WHERE E1.DNO=D1.DNO
       AND E2.DNO=D2.DNO
       AND D1.DNOM='COMMERCIAL'
       AND D2.DNOM='PRODUCTION'
       AND E1.DATEEMB=E2.DATEEMB
```

ou

```
SELECT ENOM
  FROM EMP, DEPT
 WHERE EMP.DNO=DEPT.DNO
       AND DNOM='COMMERCIAL'
       AND DATEEMB IN (SELECT DATEEMB
                         FROM EMP, DEPT
                        WHERE EMP.DNO=DEPT.DNO
                              AND DNOM='PRODUCTION');
```

Requête 18 : Donner les noms des employés embauchés avant *tous* les employés du département 1.

Solution :
Algèbre :

$$R1 := \rho_{DATEEMB/DATE1}(\pi_{DATEEMB}(\sigma_{DNO=1}(EMP)))$$
$$R2 := \pi_{ENOM}(EMP - (EMP \rhd\!\!< _{DATEEMB>=DATE1}\ R1))$$

SQL :

```
SELECT ENOM
  FROM EMP
 WHERE DATEEMB < ALL (SELECT DATEEMB
                        FROM EMP
                       WHERE DNO=1);
```

Requête 19 : Donner les noms des employés ayant le même emploi et le même directeur que JOE.

Solution :
Algèbre :

$$R1 := \pi_{DIR,PROF}(\sigma_{ENOM='JOE'}(EMP) \bowtie DEPT)$$
$$R2 := \pi_{ENOM}((EMP \bowtie DEPT) \bowtie R1)$$

– *R1(DIR,PROF): le directeur de Joe et sa profession*

– *R2(ENOM): résultat*

SQL :

```
SELECT ENOM
  FROM EMP, DEPT
 WHERE ENOM <> 'JOE'
       AND EMP.DNO = DEPT.DNO
       AND (PROF, DIR) = (SELECT PROF, DIR
                            FROM EMP, DEPT
                           WHERE ENOM='JOE'
                                 AND EMP.DNO = DEPT.DNO);
```

4.3.3 Valeurs Nulles, Tris, Groupes, Agrégats et Expressions

Requête 20 : Donner la liste des employés ayant une commission.

Solution :

```
SELECT *
  FROM EMP
 WHERE COMM IS NOT NULL;
```

Requête 21 : Donner les noms, emplois et salaires des employés par emploi croissant et, pour chaque emploi, par salaire décroissant.

Solution :

```
SELECT ENOM, PROF, SAL
  FROM EMP
 ORDER BY PROF ASC, SAL DESC;
```

Requête 22 : Donner le salaire moyen des employés.

Solution :

```
SELECT AVG(SAL) AS 'SALAIRE MOYEN'
  FROM EMP;
```

Requête 23 : Donner le nombre d'employés du département PRODUCTION.

Solution :

```
SELECT COUNT(EMP.*)
  FROM EMP, DEPT
 WHERE EMP.DNO = DEPT.DNO
       AND DEPT.DNOM = 'PRODUCTION';
```

Requête 24 : Les numéros de département et leur salaire maximum?

Solution :

```
SELECT DNO, MAX(SAL)
  FROM EMP
 GROUP BY DNO;
```

Requête 25 : Donner les noms des employés ayant le salaire maximum de chaque département.

Solution :

```
SELECT ENOM
  FROM EMP
 WHERE (DNO, SAL) IN (SELECT DNO, MAX(SAL)
                        FROM EMP
                        GROUP BY DNO);
```

ou

```
SELECT ENOM
  FROM EMP E
 WHERE SAL = (SELECT MAX(SAL)
                FROM EMP F
               WHERE F.DNO = E.DNO);
```

Requête 26 : Les professions et leur salaire moyen?

Solution :

```
SELECT PROF, AVG(SAL)
  FROM EMP
 GROUP BY PROF;
```

Requête 27 : Le salaire moyen le plus bas (par profession)?

Solution :

```
SELECT MIN(AVG(SAL))
  FROM EMP
 GROUP BY PROF;
```

Requête 28 : Donner les emplois ayant le salaire moyen le plus bas; donnez aussi leur salaire moyen.

Solution :

```
SELECT PROF, AVG(SAL)
  FROM EMP
 GROUP BY PROF
HAVING AVG(SAL) = (SELECT MIN(AVG(SAL))
                     FROM EMP
                    GROUP BY PROF);
```

Chapitre 5

Algèbre - SQL : Appartements - Écoles

5.1 Schéma

IMMEUBLE (**ADI**, NBETAGES, DATEC, PROP)
APPIM (**ADI**, **NAPR**, OCCUP, TYPE, SUPER, ETAGE)
PERSONNE (**NOM**, AGE, PROF, ADR, NAPR)
ÉCOLE (**NOMEC**, ADEC, NBCLASSES, DIR)
CLASSE (**NOMEC**, **NCL**, MAITRE, NBEL)
ENFANT (**NOMP**, **PRENOM**, AN, NOMEC, NCL)

avec la signification suivante :

1. Relation IMMEUBLE

ADI : adresse d'immeuble, clé; on fait l'hypothèse pour simplifier, que l'adresse identifie de manière unique un immeuble
NBETAGES : nombre d'étages d'un immeuble
DATEC : date de construction
PROP : nom du propriétaire de l'immeuble qui est une personne

2. Relation APPIM (Appartement)

ADI : adresse d'immeuble
NAPR : numéro d'appartement
OCCUP : occupant de l'appartement (nom de la personne)
TYPE : type de l'appartement (Studio, F2, . . .)
SUPER : superficie de l'appartement
ETAGE : étage où se situe l'appartement

3. Relation PERSONNE

NOM : nom de personne, clé; on fait l'hypothèse pour simplifier, que ce nom est unique sur l'ensemble des personnes que l'on considère dans la base
AGE : âge de la personne

PROF : profession de la personne
ADR : adresse de la résidence d'une personne, il s'agit d'un immeuble
NAPR : numéro d'appartement

4. Relation ÉCOLE

NOMEC : nom d'une école, clé
ADEC : adresse d'une école
NBCLASSES : nombre de classes
DIR : nom du directeur

5. Relation CLASSE

NOMEC : nom d'une école
NCL : nom de la classe, e.g., CP1, CE2, CE3, etc...
MAITRE : nom de l'instituteur
NBEL : nombre d'élèves dans la classe

6. Relation ENFANT

NOMP : nom de la personne responsable de l'enfant, clé e.g., père, mère etc...
PRENOM : prénom de l'enfant
AN : année de naissance
NOMEC : nom d'une école
NCL : nom de la classe

La relation **IMMEUBLE** décrit un ensemble d'immeubles. Chaque immeuble a un propriétaire. La relation **APPIM** décrit pour chaque immeuble l'ensemble des appartements qui le compose. Chaque appartement peut héberger plusieurs personnes mais il y en a une qui est responsable (par exemple le locataire) et qui est désignée par le constituant OCCUP. Si l'appartement est inoccupé, ce constituant prend la valeur NULL. La relation **PERSONNE** décrit un ensemble de personnes. ADR et NAPR représentent l'adresse où réside une personne. Une personne peut avoir plusieurs enfants décrits par la relation **ENFANT**. Pour simplifier, on ne considère que les enfants allant à l'école primaire. Les écoles et les classes sont décrites dans les relations **ÉCOLE** et **CLASSE**.

5.2 Requêtes

Exprimer les requêtes suivantes à l'aide de l'algèbre relationnelle, puis les traduire en SQL.

Requête 1 : Donner l'adresse des immeubles ayant plus de 10 étages et construits avant 1970.

Solution :

$$\pi_{ADI}(\sigma_{NBETAGES>10 \wedge DATEC<1970}\ IMMEUBLE)$$

```
SELECT ADI
FROM IMMEUBLE
WHERE NBETAGES > 10 AND DATEC < 1970
```

Requête 2 : Donner les noms des personnes qui habitent dans un immeuble dont ils sont propriétaires (occupants et habitants).

Solution :

$$\pi_{NOM}(PERSONNE \bowtie_{NOM=PROP \wedge ADR=ADI} IMMEUBLE)$$

```
SELECT NOM
  FROM PERSONNE, IMMEUBLE
 WHERE NOM = PROP AND ADR = ADI
```

Requête 3 : Donner les noms des personnes qui ne sont pas propriétaires.

Solution :

$$\pi_{NOM}(PERSONNE) - \pi_{PROP}(IMMEUBLE)$$

```
SELECT NOM
  FROM PERSONNE
MINUS
SELECT PROP
  FROM IMMEUBLE
```

ou

```
SELECT NOM
  FROM PERSONNE
 WHERE NOM NOT IN (SELECT PROP
                     FROM IMMEUBLE)
```

Requête 4 : Donner les adresses des immeubles possédés par des informaticiens dont l'âge est inférieur à 40 ans.

Solution :

$$\begin{aligned} R1 &:= \sigma_{AGE<40 \wedge PROF='INFORMATICIEN'} PERSONNE \\ R2 &:= \pi_{ADI}(R1 \bowtie_{NOM=PROP} IMMEUBLE) \end{aligned}$$

```
SELECT ADI
FROM PERSONNE, IMMEUBLE
WHERE NOM = PROP AND AGE < 40 AND PROF = 'INFORMATICIEN'
```

Requête 5 : Donner la liste des *occupants* (nom, âge, profession) des immeubles possédés par DUPONT.

Solution :

$$\pi_{NOM,AGE,PROF}(\sigma_{PROP='DUPONT'}(IMMEUBLE) \bowtie_{ADI=ADR} (PERSONNE))$$

```
SELECT P.NOM, P.AGE, P.PROF
  FROM PERSONNE P, IMMEUBLE I, APPIM A
 WHERE I.ADI = A.ADI
       AND I.PROP = 'DUPONT'
       AND P.NOM = A.OCCUP
```

Requête 6 : Donner le nom et la profession des propriétaires d'immeubles où il y a des appartements vides.

Solution :

$$
\begin{aligned}
R1 &:= \pi_{ADI,NAPR}(APPIM \bowtie_{OCCUP=NOM} PERSONNE) \\
R2 &:= \pi_{ADI,NAPR}\, APPIM - R1 \\
R3 &:= \pi_{NOM,PROF}(PERSONNE \bowtie_{NOM=PROP} (IMMEUBLE \bowtie R2))
\end{aligned}
$$

- *R1(ADI,NAPR) : adresses et numéros d'appartement occupés*
- *R2(ADI,NAPR) : adresses et numéros d'appartement vides*
- *R3(NOM,PROF) : nom et profession des propriétaires d'immeubles avec des appartements vides*

Avec valeur nulle :

```
SELECT DISTINCT P.NOM, P.PROF
  FROM APPIM A, IMMEUBLE I, PERSONNE P
 WHERE P.NOM = I.PROP AND I.ADI = A.ADI
       AND A.OCCUP IS NULL
```

Sans valeurs nulles :

```
SELECT DISTINCT P.NOM, P.PROF
  FROM APPIM A, IMMEUBLE I, PERSONNE P
 WHERE P.NOM = I.PROP
       AND I.ADI = A.ADI
       AND NOT EXISTS (SELECT *
                         FROM PERSONNE O
                        WHERE O.ADR = I.ADI
                              AND O.NAPR = A.NAPR)
```

Requête 7 : Donner les noms des maîtres qui habitent dans le même immeuble (à la même adresse) qu'au moins un de leurs élèves (on suppose que les enfants vivent sous le même toit que leur responsable).

Solution :

$$
\begin{aligned}
R1 &:= \pi_{ADR,NOMEC,NCL,MAITRE}(PERSONNE \bowtie_{MAITRE=NOM} CLASSE) \\
R2 &:= \pi_{ADR,NOMEC,NCL}(PERSONNE \bowtie_{NOMP=NOM} ENFANT) \\
R3 &:= \pi_{MAITRE}(R1 \bowtie R2)
\end{aligned}
$$

- *R1(...) : adresses des maîtres, écoles, classes et maîtres*
- *R2(...) : adresses des élèves, écoles, classes*
- *R3(MAITRE) : résultat*

ou:

$$
\begin{aligned}
R1 &:= \pi_{ADR,MAITRE}(PERSONNE \bowtie_{NOMP=NOM} (ENFANT \bowtie CLASSE)) \\
R2 &:= \pi_{MAITRE}(\sigma_{NOM=MAITRE}(R1 \bowtie PERSONNE))
\end{aligned}
$$

- *R1(ADR,MAITRE) : adresses des enfants et leurs maîtres*
- *R2(MAITRE) : résultat*

```
SELECT DISTINCT M.NOM
  FROM CLASSE C, PERSONNE M,
       ENFANT E, PERSONNE R
 WHERE C.MAITRE = M.NOM
       AND E.NOMEC = C.NOMEC
       AND E.NCL = C.NCL
       AND E.NOMP = R.NOM
       AND M.ADR = R.ADR
```

ou (imbriqué) :

```
SELECT C.MAITRE
  FROM CLASSE C, PERSONNE P
 WHERE C.MAITRE = P.NOM AND P.ADR IN (SELECT P.ADR
                                        FROM PERSONNE P, ENFANT E
                                       WHERE P.NOM = E.NOMP AND
                                             E.NOMEC = C.NOMEC AND
                                             E.NCL = C.NCL)
```

Requête 8 : Donner l'adresse de l'immeuble, la date de construction, le type d'appartement et l'étage où habitent chacun des maîtres des enfants de DUPONT.

Solution :

$$
\begin{aligned}
R1 &:= \pi_{MAITRE}(\sigma_{NOMP='DUPONT'}\ ENFANT \bowtie CLASSE) \\
R2 &:= R1 \bowtie_{MAITRE=NOM}\ PERSONNE \\
R3 &:= \pi_{ADI,DATEC,TYPE,ETAGE}(R2 \bowtie IMMEUBLE)
\end{aligned}
$$

- *R1(MAITRE) : les maîtres des enfants de DUPONT*
- *R2(...) : les noms, adresses, ... des maîtres*
- *R3(ADI,DATEC,TYPE,ETAGE) : résultat*

```
SELECT A.ADI, I.DATEC, A.TYPE, A.ETAGE
FROM CLASSE C, ENFANT E, PERSONNE P, IMMEUBLE I, APPIM A
WHERE I.ADI = P.ADR
      AND A.NAPR = P.NAPR
      AND A.ADI = I.ADI
      AND P.NOM = C.MAITRE
      AND C.NOMEC = E.NOMEC
      AND C.NCL = E.NCL
      AND E.NOMP = 'DUPONT'
```

Requête 9 : Donner le nom et l'âge des maîtres qui habitent dans un immeuble dont le propriétaire est responsable d'un de leurs élèves.

Solution :

$$
\begin{aligned}
R1 &:= \pi_{MAITRE,AGE,ADR,NOMP}(ENFANT \bowtie (CLASSE \bowtie_{MAITRE=NOM}\ PERSONNE)) \\
R2 &:= \pi_{MAITRE,AGE}(R1 \bowtie_{NOMP=PROP \wedge ADR=ADI}\ IMMEUBLE)
\end{aligned}
$$

- *R1(MAITRE,AGE,ADR,NOMP) : pour chaque enfant : le nom, l'âge et l'adresse des maîtres et le nom du responsable*
- *R2(MAITRE,AGE) : résultat*

```
SELECT M.NOM, M.AGE
FROM IMMEUBLE I, ENFANT E, CLASSE C, PERSONNE M
WHERE I.ADI = M.ADR
      AND I.PROP = E.NOMP
      AND C.NCL = E.NCL
      AND C.NOMEC = E.NOMEC
      AND M.NOM = C.MAITRE
```

Requête 10 : Donner le nom et l'âge des personnes qui sont propriétaires mais qui ne sont ni maître ni directeur d'école.

Solution :

$$
\begin{aligned}
R1 &:= \rho_{MAITRE/PROP}(\pi_{MAITRE}\ CLASSE) \\
R2 &:= \rho_{DIR/PROP}(\pi_{DIR}\ ECOLES) \\
R3 &:= \pi_{PROP}\ IMMEUBLE - (R1 \cup R2) \\
R4 &:= \pi_{AGE,NOM}(PERSONNE \bowtie_{PROP=NOM}\ R3)
\end{aligned}
$$

```
SELECT NOM, AGE
FROM PERSONNE
WHERE NOM IN (SELECT PROP
                FROM IMMEUBLE
              MINUS
              (SELECT DIR
                 FROM ECOLE
               UNION
               SELECT MAITRE
                 FROM CLASSE))
```

5.3 Mise à jour

Requête 11 : Ajouter un enfant de nom **np**, de prénom **e**, né en **a** et l'inscrire à la classe **c** de l'école **ec**.

Solution :

```
INSERT INTO ENFANT VALUE (np,e,a,ec,c);
UPDATE CLASSE
SET    NBEL = NBEL + 1
WHERE  NOMEC = ec AND NCL = c;
```

5.4 Contraintes

Indiquer de la façon la plus formelle possible certaines contraintes que les données de la base doivent respecter pour être conformes à la réalité modélisée ici.

Solution :

Inclusions des ensembles, par exemple :

- $IMMEUBLE[PROP] \subseteq PERSONNE[NOM]$
- $ECLOLE[DIR] \subseteq PERSONNE[NOM]$
- $CLASSE[MAITRE] \subseteq PERSONNE[NOM]$
- $ENFANT[NOMP] \subseteq PERSONNE[NOM]$

L'étage d'un appartement dans un immeuble est inférieur ou égal au nombre d'étages de cet immeuble :

$$(IMMEUBLE(a,n,d,p,) \wedge APPIM(a,ap,t,s,e)) \Rightarrow e \leq n$$

Le nombre d'élèves dans une classe correspond à l'ensemble des n-uplets dans la relation ENFANT :

$$CLASSE(e,c,m,n) \Rightarrow card(\{ENFANT : NOMEC = e \wedge NCL = c\}) = n$$

Pour les professions

$$\forall d \in \{ECOLE[DIR]\} \Rightarrow \exists a,ad,apPERSONNE(d,a,`DIRECTEUR',ad,ap)$$

$$\forall m \in \{CLASSE[MAITRE]\} \Rightarrow \exists a,ad,apPERSONNE(d,a,`INSTITUTEUR',ad,ap)$$

etc...

Chapitre 6

SQL : Fournisseurs - Produits - Clients

6.1 Schéma

Les exemples suivants sont tirés du livre *A Guide to DB, Third Edition* de C.J. Date.

```
CREATE TABLE FOURNISSEUR
        ( F#            CHAR(5)         NOT NULL,
          FNOM          CHAR(20)        NOT NULL WITH DEFAULT,
          STATUS        SMALLINT        NOT NULL WITH DEFAULT,
          VILLE         CHAR(15)        NOT NULL WITH DEFAULT,
        PRIMARY KEY ( F# ) );

CREATE TABLE PRODUIT
        ( P#            CHAR(6)         NOT NULL,
          PNOM          CHAR(20)        NOT NULL WITH DEFAULT,
          COULEUR       CHAR(6)         NOT NULL WITH DEFAULT,
          POIDS         SMALLINT        NOT NULL WITH DEFAULT,
        PRIMARY KEY ( P# ) );

CREATE TABLE CLIENT
        ( C#            CHAR(6)         NOT NULL,
          CNOM          CHAR(20)        NOT NULL WITH DEFAULT,
          VILLE         CHAR(15)        NOT NULL WITH DEFAULT,
        PRIMARY KEY ( C# ) );

CREATE TABLE COMMANDE
        ( F#            CHAR(5)         NOT NULL,
          P#            CHAR(6)         NOT NULL,
          C#            CHAR(6)         NOT NULL,
          QTE           SMALLINT,
        PRIMARY KEY ( F#, P#, C# ) );

CREATE UNIQUE INDEX FX ON FOURNISSEUR ( F# );
```

6.2 Requêtes

Exprimer les requêtes suivantes en SQL.

Requête 1 : Toutes les informations sur les clients.

Solution :

```
SELECT C#, CNOM, VILLE
  FROM CLIENT;
```

ou

```
SELECT *
  FROM CLIENT;
```

Requête 2 : Toutes les informations sur les clients à Paris.

Solution :

```
SELECT *
  FROM CLIENT
 WHERE VILLE = 'Paris';
```

Requête 3 : La liste triée des numéros des fournisseurs du client avec le numéro C1.

Solution :

```
SELECT DISTINCT F#
  FROM COMMANDE
 WHERE C# = 'C1'
 ORDER BY F#;
```

Requête 4 : Les commandes avec une quantité entre 300 et 750.

Solution :

```
SELECT *
  FROM COMMANDE
 WHERE QTE >= 300 AND QTE <= 750;
```

ou

```
SELECT *
  FROM COMMANDE
 WHERE QTE BETWEEN 300 AND 750;
```

Requête 5 : Les commandes avec une quantité différente de NULL.

Solution :

```
SELECT *
  FROM COMMANDE
 WHERE QTE IS NOT NULL;
```

ou

```
SELECT *
  FROM COMMANDE
 WHERE QTE = QTE;
```

Requête 6 : Les numéros des clients qui sont situés dans une ville qui commence par “P”.

Solution :

```
SELECT C#
  FROM CLIENTS
 WHERE VILLE LIKE 'P%';
```

ou

```
SELECT C#
  FROM CLIENTS
 WHERE SUBSTR (VILLE, 1, 1) = 'P';
```

Requête 7 : Les numéros des fournisseurs et des clients qui sont situés dans la même ville.

Solution :

```
SELECT F#, C#
  FROM FOURNISSEUR, CLIENT
 WHERE FOURNISSEUR.VILLE = CLIENT.VILLE;
```

Requête 8 : Les numéros des fournisseurs et des clients qui ne sont pas situés dans la même ville.

Solution :

```
SELECT F#, C#
  FROM FOURNISSEUR, CLIENT
 WHERE NOT FOURNISSEUR.VILLE = CLIENT.VILLE;
```

ou

```
SELECT F#, C#
  FROM FOURNISSEUR, CLIENT
 WHERE FOURNISSEUR.VILLE <> CLIENT.VILLE;
```

Requête 9 : Les numéros des produits fournis par des fournisseurs Parisiens.

Solution :

```
SELECT DISTINCT P#
  FROM COMMANDE, FOURNISSEUR
 WHERE COMMANDE.F# = FOURNISSEUR.F#
   AND VILLE = 'Paris';
```

Requête 10 : Les numéros des produits fournis par des fournisseurs Parisiens à des clients Marseillais.

Solution :

```
SELECT DISTINCT P#
  FROM COMMANDE, FOURNISSEUR, CLIENT
 WHERE COMMANDE.F# = FOURNISSEUR.F#
   AND COMMANDE.C# = CLIENT.C#
   AND FOURNISSEUR.VILLE = 'Paris'
   AND CLIENT.VILLE = 'Marseille';
```

Requête 11 : Les couples de villes (v_i, v_j) tel qu'il existe au moins un fournisseur dans la ville v_i d'un client dans la ville v_j.

Solution :

```
SELECT DISTINCT FOURNISSEUR.VILLE, CLIENT.VILLE
  FROM COMMANDE, FOURNISSEUR, CLIENT
 WHERE COMMANDE.F# = FOURNISSEUR.F#
   AND COMMANDE.C# = CLIENT.C#;
```

Requête 12 : Les numéros des produits fournis à des clients situés dans la même ville que leurs fournisseurs.

Solution :

```
SELECT DISTINCT COMMANDE.P# FROM COMMANDE, FOURNISSEUR, CLIENT
  WHERE COMMANDE.F# = FOURNISSEUR.F# AND COMMANDE.C# = CLIENT.C#
    AND CLIENT.VILLE = FOURNISSEUR.VILLE;
```

Requête 13 : Les numéros des clients qui ont au moins un fournisseur situé dans une autre ville.

Solution :

```
SELECT DISTINCT CLIENT.C#
  FROM COMMANDE, FOURNISSEUR, CLIENT
 WHERE COMMANDE.F# = FOURNISSEUR.F#
   AND COMMANDE.C# = CLIENT.C#
   AND CLIENT.VILLE <> FOURNISSEUR.VILLE;
```

Requête 14 : Les couples de produits qui sont fournis par le même fournisseur.

Solution :

```
SELECT DISTINCT A.P#, B#.P#
  FROM COMMANDE A, COMMANDE B
 WHERE A.F# = B.F#
   AND A.P# > B.P#;
```

Chapitre 7

Calcul - SQL - Algèbre : Cinémas - Films

7.1 Schéma

Les exemples suivants sont tirés du livre *Foundations of Databases* de S. Abiteboul, R. Hull et V. Vianu.

SALLE (Nom,Horaire,Titre)
FILM (Titre, Realisateur, Acteur)
PRODUIT (Producteur,Titre)
VU (Spectateur,Titre)
AIME (Spectateur,Titre)

Un film est réalisé par un metteur en scène mais peut être financé par plusieurs Producteurs. Un Spectateur peut aimer un film sans l'avoir vu.

7.2 Requêtes

Écrire les requêtes suivantes en algèbre relationnel, en calcul à variable n-uplet et en calcul à variable domaine.

7.2.1 Interrogation d'une seule Relation

Requête 1 : Dans quelle salle et à quelle heure peut on voir le film "Mad Max"?

Solution :
Algèbre :

$$\pi_{Nom,Horaire}(\sigma_{Titre='Mad\ Max'}(SALLE))$$

SQL :

```
SELECT Nom, Horaire
  FROM SALLE
 WHERE Titre = 'Mad Max'
```

Calcul n-uplet :

$$\{sx.Nom,sx.Horaire \mid SALLE(sx) \wedge sx.Titre =' Mad\ Max'\}$$

Calcul domaine :

$$\{nx,hx \mid SALLE('Mad\ Max',nx,hx)\}$$

Requête 2 : Quels sont les films réalisés par Orson Welles ?

Solution :
Algèbre :

$$\pi_{Titre}(\sigma_{Realisateur='Welles'}(FILM))$$

SQL :

```
SELECT Titre
  FROM FILM
 WHERE Realisateur = 'Welles'
```

Calcul n-uplet :

$$\{fx.Titre \mid FILM(fx) \wedge fx.Realisateur =' Welles'\}$$

Calcul domaine :

$$\{tx \mid \exists ax FILM(tx,'Welles',ax)\}$$

Requête 3 : Quels sont les Acteurs du film “Ran” ?

Solution :
Algèbre :

$$\pi_{Acteur}(\sigma_{Titre='Ran'}(FILM))$$

SQL :

```
SELECT Acteur
  FROM FILM
 WHERE Titre = 'Ran'
```

Calcul n-uplet :

$$\{fx.Acteur \mid FILM(fx) \wedge fx.Titre =' Ran'\}$$

Calcul domaine :

$$\{ax \mid \exists rx FILM('Ran',rx,ax)$$

7.2.2 Jointures

Requête 4 : Dans quelles salles peut-on voir un film avec Simone Signoret ?

Solution :
Algèbre :

$$\pi_{Nom}(SALLE \bowtie (\sigma_{Acteur='Signoret'}(FILM)))$$

SQL :

```
SELECT Nom
  FROM SALLE, FILM
 WHERE SALLE.Titre = FILM.Titre
       AND FILM.Acteur = 'Signoret'
```

Calcul n-uplet :

$$\{sx.Nom \mid \exists fx(SALLE(sx) \wedge FILM(fx) \wedge fx.Acteur =' Signoret' \wedge sx.Titre = fx.Titre)\}$$

Calcul domaine :

$$\{nx \quad | \quad \exists tx,rx,hx(FILM(tx,rx,'Signoret') \wedge SALLE(nx,hx,tx))\}$$

Requête 5 : Dans quelles salles peut on voir Marlon Brando après 16h ?

Solution :
Algèbre :

$$\pi_{Nom}(\sigma_{Horaire>16}(SALLE) \bowtie (\sigma_{Acteur=' Brando'}(FILM)))$$

SQL :

```
SELECT Nom
  FROM SALLE, FILM
 WHERE SALLE.Titre = FILM.Titre
       AND FILM.Acteur = 'Brando'
       AND SALLE.Horaire > 16
```

Calcul n-uplet :

$$\{sx.Nom \quad | \quad \exists fx(SALLE(sx) \wedge FILM(fx) \wedge fx.Acteur =' Brando' \wedge sx.Horaire > 16 \wedge sx.Titre = fx.Titre)\}$$

Calcul domaine :

$$\{nx \mid \exists tx,rx,hx(FILM(tx,rx,'Brando') \wedge SALLE(nx,hx,tx) \wedge hx > 16)\}$$

Requête 6 : Quels sont les Acteurs qui ont produit un film ?

Solution :
Algèbre :

$$\pi_{Acteur}(FILM \bowtie_{Acteur=Producteur} PRODUIT)$$

SQL :

```
SELECT Acteur
  FROM PRODUIT, FILM
 WHERE Acteur = Producteur
```

Calcul n-uplet :

$$\{fx.Acteur \quad | \quad \exists px(PRODUIT(px) \wedge FILM(fx) \wedge fx.Acteur = px.Producteur)\}$$

Calcul domaine :

$$\{ax \mid \exists tx,ty,rx(FILM(tx,rx,ax) \wedge PRODUIT(ax,ty))\}$$

Requête 7 : Quels sont les Acteurs qui ont produit un film dans lequel ils jouent ?

Solution :

Algèbre :

$$\pi_{Acteur}(\sigma_{Acteur=Producteur}(FILM \bowtie PRODUIT))$$

SQL :

```
SELECT Acteur
  FROM FILM, PRODUIT
 WHERE FILM.Titre = PRODUIT.Titre
       AND Acteur = Producteur
```

Calcul n-uplet :

$$\{fx.Acteur \mid \exists px(PRODUIT(px) \wedge FILM(fx) \wedge fx.Acteur = px.Producteur \wedge fx.Titre = px.Titre)\}$$

Calcul domaine :

$$\{ax \mid \exists tx,rx(FILM(tx,rx,ax) \wedge PRODUIT(ax,tx))\}$$

Requête 8 : Quels sont les Acteurs qui ont produit et réalisé un film ?

Solution :
Algèbre :

$$R1 := \pi_{Realisateur}(\sigma_{Producteur=Realisateur}(PRODUIT \bowtie FILM))$$
$$R2 := \pi_{Acteur}(FILM \bowtie_{Acteur=Realisateur} R1)$$

SQL :

```
SELECT A.Acteur
  FROM FILM A, FILM B, PRODUIT C
 WHERE A.Acteur = B.Realisateur
       AND B.Realisateur = C.Producteur
       AND B.Titre = C.Titre
```

Calcul n-uplet :

$$\{fx.Acteur \mid \exists fy,px(FILM(fx) \wedge FILM(fy) \wedge PRODUIT(px) \wedge fx.Acteur = px.Producteur \wedge fy.Realisateur = fx.Acteur \wedge px.Titre = fy.Titre)\}$$

Calcul domaine :

$$\{ax \mid \exists tx,ty,rx,ay(FILM(tx,rx,ax) \wedge PRODUIT(ax,ty) \wedge FILM(ty,ax,ay))\}$$

Requête 9 : Quels sont les Producteurs qui regardent les films qu'ils ont produits ?

Solution :
Algèbre :

$$\pi_{Producteur}(\sigma_{Spectateur=Producteur}(PRODUIT \bowtie VU))$$

Calcul n-uplet :

$$\{px.Producteur \mid \exists vx(PRODUIT(px) \wedge VU(vx) \wedge px.Producteur = vx.Spectateur \wedge px.Titre = vx.Titre)\}$$

Calcul domaine :

$$\{px \mid \exists tx(PRODUIT(px,tx) \wedge VU(px,tx))\}$$

7.2.3 Difference

Requête 10 : Quels films ne passent en ce moment dans aucune salle ?

Solution :
Algèbre :

$$\pi_{Titre}(FILM) - \pi_{Titre}(SALLE)$$

Calcul n-uplet :

$$\{fx.Titre \mid FILM(fx) \wedge \neg\exists sx(SALLE(sx) \wedge sx.Titre = fx.Titre)\}$$

Calcul domaine :

$$\{tx \mid \exists rx,axFILM(tx,rx,ax) \wedge \neg\exists nx,hx(SALLE(nx,hx,tx))\}$$

Requête 11 : Quel est le résultat de la requête suivante (ATTENTION, la requête n'est pas saine!) ?

$$\{fx.Titre \mid FILM(fx) \wedge \forall sx(SALLE(sx) \wedge sx.Titre \neq fx.Titre)\}$$

Solution :
La requête n'est pas indépendante du domaine : la variable sx peut correspondre à toutes les salles, mais aussi à toutes les acteurs, films etc.... Dans le dernier cas, le résultat est vide.

Requête 12 : Quels Spectateurs aiment un film qu'ils n'ont pas vu ?

Solution :
Algèbre :

$$\pi_{Spectateur}(AIME - VU)$$

Calcul n-uplet :

$$\{ax.Spectateur \mid AIME(ax) \wedge \neg\exists vx(VU(vx) \wedge ax.Spectateur = vx.Spectateur \wedge ax.Titre = vx.Titre)\}$$

Calcul domaine :

$$\{sx \mid \exists txAIME(sx,tx) \wedge \neg VU(sx,tx)\}$$

Requête 13 : Qui n'aime aucun film qu'il a vu ?

Solution :
Algèbre :

$$\pi_{Spectateur}(VU) - \pi_{Spectateur}(AIME \bowtie VU)$$

Calcul n-uplet :

$$\{vx.Spectateur \mid VU(vx) \wedge \neg\exists ax,vz(AIME(ax) \wedge VU(vz) \wedge ax.Spectateur = vx.Spectateur \wedge ax.Titre = vz.Titre \wedge vx.Spectateur = vz.Spectateur)\}$$

Calcul domaine :

$$\{sx \mid VU(sx,tx) \wedge \neg\exists ty(AIME(sx,ty) \wedge VU(sx,ty))\}$$

Requête 14 : Qui n'a produit aucun film de Doillon ?

Solution :
Algèbre :

$$\pi_{Producteur}(PRODUIT) - \pi_{Producteur}(PRODUIT \bowtie (\sigma_{Realisateur='Doillon'} FILM))$$

Calcul n-uplet :

$$\{px.Producteur \mid PRODUIT(px) \wedge \neg\exists px', fx(PRODUIT(px') \wedge FILM(fx) \wedge fx.Realisateur =' Doillon' \wedge fx.Titre = px'.Titre \wedge px'.Producteur = px.Producteur)\}$$

Calcul domaine :

$$\{px \mid \exists tx PRODUIT(px,tx) \wedge \neg\exists ax,tx'(PRODUIT(px,tx') \wedge FILM(tx,'Doillon',ax))\}$$

Requête 15 : Qui a produit un film qui ne passe dans aucune salle ?

Solution :
Algèbre :

$$\pi_{Producteur}(PRODUIT \bowtie (\pi_{Titre} FILM - \pi_{Titre} SALLE))$$

Calcul n-uplet :

$$\{px.Producteur \mid PRODUIT(px) \wedge \exists fx(FILM(fx) \wedge fx.Titre = px.Titre \wedge \neg\exists sx(SALLE(sx) \wedge sx.Titre = fx.Titre))\}$$

Calcul domaine :

$$\{px \mid \exists tx PRODUIT(px,tx) \wedge \exists rx,ax(FILM(tx,rx,ax) \wedge \neg\exists nx,hx(SALLE(nx,hx,tx)))\}$$

7.2.4 Division

Requête 15 : Quels Spectateurs ont vu tous les films ? (ou Spectateurs pour lesquels il n'existe pas un film qu'ils n'ont pas vu)

Solution :
Algèbre :

$$VU \div \pi_{Titre}(FILM)$$

Calcul n-uplet :

$$\{vx.Spectateur \mid VU(vx) \wedge \neg\exists fx(FILM(fx) \wedge \neg\exists vy(VU(vy) \wedge vy.Titre = fx.Titre \wedge vy.Spectateur = vx.Spectateur))\}$$

Calcul domaine :

$$\{sx \mid \exists tx VU(sx,tx) \wedge \neg\exists ty(\exists rx,ax(FILM(ty,rx,ax) \wedge \neg VU(sx,ty)))\}$$

Attention, la requête suivante n'est pas saine : la variable tx peut correspondre aux titre de films mais aussi aux acteurs, spectateurs etc... Dans le dernier cas, le résultat est toujours vide.

$$\{sx \mid \forall tx(\exists rx,ax(FILM(tx,rx,ax) \wedge VU(sx,tx)))\}$$

Requête 16 : Quels Acteurs jouent dans tous les films de Welles ? (ou Acteurs pour lesquels il n'existe pas un film de Welles qu'ils n'ont pas joué)

Solution :
Algèbre :

$$(\pi_{Acteur,Titre}(FILM)) \div (\pi_{Titre}(\sigma_{Realisateur='Welles'}(FILM)))$$

SQL :

```
SELECT fx.Acteur
  FROM FILM fx
 WHERE NOT EXISTS ( SELECT fy
                      FROM FILM fy
                     WHERE fy.Realisateur = 'Welles'
                       AND NOT EXISTS ( SELECT fz
                                          FROM FILM fz
                                         WHERE fz.Titre = fy.Titre
                                           AND fz.Acteur = fx.Acteur
                                         ) )
```

Calcul n-uplet :

$$\{fx.Acteur \mid FILM(fx) \wedge \neg\exists fy(FILM(fy) \wedge fy.Realisateur =' Welles' \wedge \neg\exists fz(FILM(fz) \wedge fz.Titre = fy.Titre \wedge fz.Acteur = fx.Acteur))\}$$

Calcul domaine :

$$\{ax \mid \exists tx,rx FILM(tx,rx,ax) \wedge \neg\exists ty(\exists ay FILM(ty,'Welles',ay)) \wedge \neg\exists ry FILM(ty,ry,ax)\}$$

Attention, la requête suivante n'est pas saine (indépendante du domaine) : elle donne le bon résultat, si ty correspond à tous les films d'Orson Welles et faux sinon.

$$\{ax \mid \forall ty FILM(ty,'Welles',ax)\}$$

Requête 17 : Quels sont les Spectateurs qui aiment tous les films qu'ils ont vu ? (ou Spectateurs pour lesquels il n'existe pas un film qu'ils ont vu et qu'ils n'ont pas aimé)

Solution :
Algèbre :

$$(AIME \bowtie \rho_{Spectateur/Spectateur'}(VU)) \div VU$$

Calcul n-uplet :

$$\{ax.Spectateur \mid AIME(ax) \wedge \neg\exists vx(VU(vx) \wedge \neg\exists ay(AIME(ay) \wedge ay.Titre = vx.Titre \wedge ay.Spectateur = ax.Spectateur))\}$$

Calcul domaine :

$$\{sx \mid \exists tx AIME(sx,tx) \wedge \neg\exists ty(VU(sx,ty) \wedge \neg AIME(sx,ty))\}$$

Requête 18 : Quels sont les Producteurs qui voient tous les films qu'ils ont produit ? (ou Producteurs pour lesquels il n'existe pas un film qu'ils ont produit et qu'ils n'ont pas vu)

Solution :

Algèbre :

$$(\pi_{Producteur}\ PRODUIT \bowtie_{Producteur=Spectateur}\ VU) \div PRODUIT$$

Calcul n-uplet :

$$\{px.Producteur \mid PRODUIT(px) \wedge \neg\exists vx(VU(vx) \wedge px.Producteur = vx.Spectateur \wedge \neg\exists py(PRODUIT(py) \wedge py.Titre = vx.Titre \wedge py.Producteur = px.Producteur))\}$$

Calcul domaine :

$$\{px \mid \exists tx PRODUIT(px,tx) \wedge \neg\exists ty(VU(px,ty) \wedge \neg PRODUIT(px,ty))\}$$

Requête 19 : Quels Producteurs voient tous les films de Kurosawa ? (ou Producteurs pour lesquels il n'existe pas un film de Kurosawa qu'ils n'ont pas vu)

Solution :
Algèbre :

$$\begin{aligned} R1 &:= \pi_{Titre}(\sigma_{Realisateur='Kurosawa'}FILM) \\ R2 &:= VU \rtimes_{Spectateur=Producteur} PRODUIT \\ R3 &:= \pi_{Spectateur}(R2 \div R1) \end{aligned}$$

- *R1(Titre) : tous les Titres de films de Kurosawa*
- *R2(...) : les films vus par des Producteurs*
- *R3(Spectateur) : résultat*

Calcul n-uplet :

$$\{px.Producteur \mid pxPRODUIT(px) \wedge \neg\exists fx(FILM(fx) \wedge fx.Realisateur =' Kurosawa' \wedge \neg\exists vx(VU(vx) \wedge vx.Titre = fx.Titre \wedge vx.Spectateur = px.Producteur))\}$$

Calcul domaine :

$$\{px \mid \exists tx PRODUIT(px,tx) \wedge \neg\exists(ty\exists ax(FILM(ty,'Kurosawa',ax)) \wedge \neg VU(px,ty))\}$$

Chapitre 8

Décomposition

8.1 Calcul des Clés

Exercice A : Soit le schéma relationnel $\mathcal{R}(A,B,C,D)$ avec les dépendance fonctionnelles $\mathcal{F} = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$. Quelles sont les clés?

Solution :

1. *B n'apparaît dans le membre droit d'aucune dépendance, donc B appartient à la clé.*
2. *Montrons que B n'est pas une clé:*

$$B^+ = B^1 = \{B,D\} \neq U$$

3. *Essayons de montrer que AB, BC, et BD sont de clés:*
   Etape 1 :

$$AB^1 = \{A,B\} \cup \{C,D\} = U$$

   En effet C vient de $AB \rightarrow C$, D vient de $B \rightarrow D$.

$$AB^+ = AB^1 = U$$

   – donc AB est une superclé.
   Etape 2 :

$$BC^1 = \{B,C\} \cup \{A,D\} = U$$

   En effet A vient de $BC \rightarrow A$, D vient de $B \rightarrow D$.

$$BC^+ = BC^1 = U$$

   – donc BC est une superclé.
   Etape 3 :

$$BD^1 = \{B,D\} \cup \{D\} = BD^+ \neq U$$

   – donc BD n'est pas une superclé.
4. *Donc, comme*
   (a) *B doit apparaître dans toute clé*
   (b) *B n'est pas une clé*

(c) AB, BC sont des superclés
alors AB et BC sont les 2 seules clés.

Exercice B : Soit le schéma relationnel $\mathcal{R}(A,B,C,D)$, $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, A \rightarrow D, D \rightarrow C\}$. Montrer que A est la seule clé.

Solution :

1. *A appartient à toute clé parce que A n'apparaît dans aucun membre droit de dépendance.*
2. $$A^1 = \{A\} \cup \{B,C\} = \{A,B,D\}$$
$$A^+ = A^1 \cup \{C\} = U$$
Donc A est une clé.
3. *de 1) et du fait que tout sous-ensemble V de U contenant une clé ne peut être qu'une superclé et non pas une clé, on en déduit que A est la seule clé.*

Exercice C : Soit le schéma relationnel $\mathcal{R}(A,B,C,D)$ et $\mathcal{F} = \{A \rightarrow B, B \rightarrow C, D \rightarrow B\}$. Trouver les clés.

Solution :

1. *A et D n'apparaissent dans aucun membre droit de dépendance et font partie de toute clé.*
2. *Si AD est une clé, alors c'est la seule*
$$AD^1 = \{A,D\} \cup \{B\}$$
$$AD^+ = AD^2 = AD^1 \cup \{C\} = U$$
Donc AD est la seule clé.

Exercice D : Soit le schéma relationnel

$$\mathcal{R}(\mathbf{C}ours, \mathbf{P}rofesseur, \mathbf{H}oraire, \mathbf{S}alle, \mathbf{E}tudiant, \mathbf{N}ote)$$
$$\mathcal{F} = \{C \rightarrow P, HS \rightarrow C, HP \rightarrow S, CE \rightarrow N, HE \rightarrow S\}$$

Montrer que la seule clé est HE.

Solution :

Etape 1 : H et E sont les 2 attributs qui n'apparaissent dans aucun membre de droite de dépendance.
Etape 2 :
$$HE^1 = \{H,E\} \cup \{S\} = \{H,E,S\}$$
S vient de $HE \rightarrow S$.
$$HE^2 = \{H,E,S\} \cup \{C\} = \{H,E,S,C\}$$
C vient de $HS \rightarrow C$.
$$HE^3 = \{H,E,S,C\} \cup \{P,N\}$$
P vient de $C \rightarrow P$, N vient de $CE \rightarrow N$.
$$HE^+ = HE^3 = U$$
Donc HE est une clé – de 1) c'est la seule clé.

Exercice E : Soit le schéma relationnel $\mathcal{R}(A,B,C,D)$, $\mathcal{F} = \emptyset$. Quelles sont les clés?

Solution :
Étant donné qu'il n'y a pas de dépendances fonctionnelles, aucun attribut dans l'univers apparaît du coté droite d'une dépendance fonctionnelle. Donc la seule clé est composée de tous les attributs dans $\mathcal{R}$.

8.2 Troisième Forme Normale

Exercice A : Est-ce que le schéma

$$\mathcal{R}(\mathbf{R}ue,\mathbf{V}ille,\mathbf{C}odePostal)$$
$$\mathcal{F} = \{RV \rightarrow C,C \rightarrow V\}$$

est en 3FN?
Définition 3FN: $\forall A \rightarrow X \in F$: A est une superclé **ou** X appartient à une clé.

Solution :

1. *Les clés sont RV et CR.*
2. *Montrons que pour toute dépendance de F, $A \rightarrow X$, soit A est une superclé, soit X appartient à une clé:*
   - *$RV \rightarrow C$: RV est une clé.*
   - *$C \rightarrow V$: V appartient à la clé RV.*

   Donc $\mathcal{R}$ est en 3FN. Donc il n'est pas nécessaire de décomposer R. Remarquez toutefois que certaines redondances ne sont pas éliminées dans une relation 3FN:

R

Rue	*Ville*	*Code-Postal*
De Gaulle	*Paris*	*75008*
Champs Elysées	*Paris*	*75008*

Non seulement on a la redondance $Ville, CodePostal$, mais on ne peut pas insérer un nouveau couple $(Ville,CodePostal)$ sans connaître une rue avec ce code.

Exercice B : Montrer que les schémas suivants ne sont pas en 3FN:
Schéma 1 :

$$\mathcal{R}(A,B,C,D)$$
$$\mathcal{F} = \{AB \rightarrow C,B \rightarrow D,BC \rightarrow A\}$$

Solution :
Clés: AB et BC (voir exercice plus haut). Dans la dépendance $B \rightarrow D$, B n'est pas une superclé et D n'appartient à aucune clé. Donc le schéma n'est pas en 3FN.

Schéma 2 :

$$\mathcal{R}(A,B,C,D)$$
$$\mathcal{F} = \{A \rightarrow B,B \rightarrow C,A \rightarrow D,D \rightarrow C\}$$

Solution :
Clé: A (voir exercice plus haut). Dans la dépendance $B \rightarrow C$, B n'est pas une superclé et C n'appartient pas à une clé (de même pour la dépendance $D \rightarrow C$. Donc le schéma n'est pas en 3FN.

Schéma 3 :

$$\mathcal{R}(C,P,H,S,E,N)$$
$$\mathcal{F} = \{C \rightarrow P,HS \rightarrow C,HP \rightarrow S,CE \rightarrow N,HE \rightarrow S\}$$

Solution :
Aucune dépendance sauf la dernière satisfait les critères de 3FN. Donc ce schéma n'est pas en 3FN.

Schéma 4 :

$$\mathcal{R}(F,A,N,P)$$

$$\mathcal{F} = \{F \rightarrow A, FN \rightarrow P\}$$

Solution :
FN est la seule clé (montrez le). Dans la dépendance $F \rightarrow A$, F n'est pas une superclé et A n'appartient pas à une clé.

Schéma 5 :

$$\mathcal{R}(M,A,D,R)$$

$$F = \{MA \rightarrow D, MD \rightarrow R\}$$

Solution :
MA est la seule clé (montrez-le). Dans la dépendance $MD \rightarrow R$, MD n'est pas une superclé et R n'appartient pas à la clé.

8.3 Décomposition sans Perte d'Information

Exercice A : Soit le schéma relationnel $\mathcal{R}(A,B,C,D)$, $\mathcal{F} = \{A \rightarrow B, C \rightarrow D\}$ et la décomposition $\Delta = \{AB, CD\}$.

1. Montrer que Δ n'est pas une décomposition sans perte d'information.
2. Donner une décomposition sans perte d'information.

Solution :

1. *Une décomposition $\{R1, R2\}$ est sans perte d'information, si on a l'une des dépendances:*

$$R1 \cap R2 \rightarrow R1 - R2$$

$$R1 \cap R2 \rightarrow R2 - R1$$

or $R1 \cap R2 = \emptyset$, donc Δ est avec perte d'information.
Exemple

R

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a2	b2	c1	d1

R1

A	B
a1	b1
a2	b2

R2

C	D
c1	d1
c2	d1

R1 × R2

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d1
a2	b2	c1	d1
a2	b2	c2	d1

Une décomposition est sans perte d'information, si par jointure (ou produit cartésien) des 2 relations $R1$ et $R2$, on peut obtenir la relation initiale R.
La décomposition précédente donne après produit cartésien une relation différente de la relation initiale.

2. $\Delta' = \{AB,ACD\}$ *est une décomposition sans perte d'information. En effet:*

$$R1 \cap R2 = A$$

$$R1 - R2 = B$$

or on a $A \rightarrow B$.
Sur l'exemple précédent:

R1

A	*B*
a1	*b1*
a2	*b2*

R2

A	*C*	*D*
a1	*c1*	*d1*
a1	*c2*	*d1*
a2	*c1*	*d1*

$$R1 \bowtie R2 = R$$

$\Delta'' = \{ABC,CD\}$ *est également une décomposition sans perte d'information:*

$$R1 \cap R2 = C$$

$$R2 - R1 = D$$

$$C \rightarrow D \in F$$

Exercice B : Soit le schéma relationnel

$$\mathcal{R}(\mathbf{F}ournisseur,\mathbf{A}dresse,\mathbf{N}omProd,\mathbf{P}rix)$$

$$\mathcal{F} = \{F \rightarrow A,FN \rightarrow P\}$$

Trouver une décomposition sans perte d'information.

Solution :

Décomposition 1 :

$$\Delta_1 = \{FA,FNP\}$$

$$R1 \cap R2 = F$$

$$R1 - R2 = A$$

$$F \rightarrow A \in \mathcal{F}$$

Décomposition 2 :

$$\Delta_2 = \{FNA,FNP\}$$

$$R1 \cap R2 = FN$$

$$R2 - R1 = P$$

$$FN \rightarrow P \in \mathcal{F}$$

Exercice C : Soit le schéma relationnel

$$\mathcal{R}(\mathbf{M}agasin,\mathbf{A}rticle,\mathbf{D}partement,\mathbf{R}esponsable)$$

$$\mathcal{F} = \{MA \rightarrow D,MD \rightarrow R\}$$

1. Montrer que la seule clé est MA.
2. Trouver une décomposition sans perte d'information.

Solution :

1. *M et A n'apparaissent pas à droite d'une dépendance. Donc toute clé inclut MA. Prouvons que MA est une clé:*

$$MA^1 = \{M,A,D\}$$
$$MA^+ = MA^2 = U$$

   Donc MA est la seule clé.
2. *$\Delta_1 = \{MAD,MDR\}$ est une décomposition sans perte d'information:*

$$R1 \cap R2 = MD$$
$$R2 - R1 = R$$
$$MD \rightarrow R \in F$$

   De même pour $\Delta_2 = \{MAD,MAR\}$:

$$R1 \cap R2 = MA$$
$$R1 - R2 = D$$
$$MA \rightarrow D \in F$$

8.4 Préservation des Dépendances Fonctionnelles

Exercice A : Soit le schéma

$$\mathcal{R}(ABCD)$$
$$\mathcal{F} = \{AB \rightarrow C,B \rightarrow D,C \rightarrow A\}$$

1. Donner une décomposition qui préserve les dépendances de $\mathcal{F}$.
2. Est-ce que cette décomposition est sans perte d'information?

Solution :

1. *On crée une relation de schéma (XA) par dépendance $X \rightarrow A$. On obtient donc la décomposition: (ABC,BD,CA). Les dépendances sont préservées. En effet*
   - *(ABC) a pour dépendance $AB \rightarrow C$ (projetée de $\mathcal{F}$ sur ABD).*
   - *(BD) a pour dépendance $B \rightarrow D$ (projetée de $\mathcal{F}$ sur BD).*
   - *(CA) a pour dépendance $C \rightarrow A$ (projetée de $\mathcal{F}$ sur CA).*

   Cependant, en créant (ABC) et (CA), on a une redondance inutile (CA est inclue dans ABC). La relation (ABC) avec les dépendances

$$\{AB \rightarrow C,C \rightarrow A\}$$

   est suffisante. La décomposition finale est donc : $\Delta = (ABC,BD)$.
2. *La décomposition Δ est sans perte d'information si :*

$$ABC \cap BD \rightarrow ABC - BD(BD - ABC).$$

   En effet,

$$B \rightarrow D \in \mathcal{F}.$$

   Donc Δ c'est une bonne décomposition c.a.d. sans perte d'information et qui préserve les dépendances. On peut vérifier que les relations (ABC) et (BD) ainsi obtenues sont en 3FN.

Exercice B : Soit le schéma

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B,B \rightarrow C,D \rightarrow B\}$$

et la décomposition

$$\Delta = (ACD,BD)$$

1. Est-ce que Δ préserve les dépendances de $\mathcal{F}$?
2. Est-ce que les relations (ACD) et (BD) obtenues sont en 3FN?

Solution :

1. *Une dépendance projetée de $\mathcal{F}^+$ sur (ACD) est $A \rightarrow C$. En effet, $A \rightarrow B$,$B \rightarrow C$ appartiennent à F. Donc par transitivité $A \rightarrow C \in \mathcal{F}^+$. De même on peut vérifier que $D \rightarrow C$ est aussi une dépendance projetée de $\mathcal{F}^+$ sur (ACD) telle que :*

   $$\{B \rightarrow C,D \rightarrow B\} \models D \rightarrow C.$$

   De plus, $D \rightarrow B$ est une dépendance projetée de $\mathcal{F}$ sur (BD). Par contre $A \rightarrow B$ et $B \rightarrow D$ ne peuvent être des dépendances projetées de $\mathcal{F}$ ni sur (ACD) ni sur (BD). La décomposition Δ ne préserve donc pas les dépendances de $\mathcal{F}$. Elle le ferait si l'ensemble des dépendances de (ACD) et de (BD) :

   $$\mathcal{G} = \{A \rightarrow C,D \rightarrow C,D \rightarrow B\}$$

   impliquerait les mêmes dépendances que $\mathcal{F}$: $\mathcal{G}^+ = \mathcal{F}^+$.
   On peut illustrer sur cet exemple pourquoi il est souhaitable qu'une décomposition préserve les dépendances : lorsqu'on insère un n-uplet bd dans la relation (BD), on vérifie qu'il satisfasse la dépendance $D \rightarrow B$. Mais on ne peut pas vérifier la contrainte $B \rightarrow C$ sans faire la jointure entre les deux relations (ACD) et (BD).
2. *La relation $R1(ACD)$ a pour dépendances $\{A \rightarrow C,D \rightarrow C\}$. La clé unique est AD. Montrons que pour toute dépendance de F, $A \rightarrow X$, soit A est une superclé, soit X appartient à une clé:*
   - *$A \rightarrow C$: C n'appartient pas à la clé.*
   - *$D \rightarrow C$: C n'appartient pas à la clé.*

   Donc $R1$ n'est pas en 3FN.
   La relation $R2(BD)$ a pour dépendance $\{D \rightarrow B\}$. La clé unique est D. Dans cette dépendance D est la clé. Donc $R2$ est en 3FN.

Exercice C : Trouver des décompositions en relations qui sont en 3FN, décompositions qui préservent les dépendances et qui sont sans perte d'information, pour les schémas suivants :

Schéma 1 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B,B \rightarrow C,A \rightarrow D,D \rightarrow C\}$$

Schéma 2 :

$$\mathcal{R}(CPHSEN)$$

$$\mathcal{F} = \{C \rightarrow P,HS \rightarrow C,HP \rightarrow S,CE \rightarrow N,HE \rightarrow S\}$$

Schéma 3 :

$$\mathcal{R}(FANP)$$

$$\mathcal{F} = \{F \rightarrow A,FN \rightarrow P\}$$

Schéma 4 :

$$\mathcal{R}(MADR)$$

$$\mathcal{F} = \{MA \rightarrow D, MD \rightarrow R\}$$

Schéma 5 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{ABC \rightarrow D\}$$

Schéma 6 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$$

Schéma 7 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B\}$$

Schéma 8 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B, B \rightarrow A, B \rightarrow D\}$$

Solution :

Rappelons que la bonne décomposition peut toujours [1] *être obtenue en :*

- *créant une relation par dépendance :*
  Si $X \rightarrow A$ est une dépendance, on crée une relation de schéma (XA) : c'est la décomposition qui préserve les dépendances.
- *si aucune clé n'est déjà incluse dans une relation créée précédemment, il faut ajouter une relation de schéma (y), où y est une clé.*
- *en remplaçant les schémas $(XA_1), \ldots, (XA_n)$ obtenus dans la première étape par un seul schéma $(XA_1 \ldots A_n)$. En effet pour chaque schéma (XA_i) on a la dépendance projetée $X \rightarrow A_i$ et la relation $(XA_1 \ldots A_n)$ préserve toutes les dépendances projetées $X \rightarrow A_1, \ldots, X \rightarrow A_n$. Dans cette cas X c'est la clé de la relation. Donc cette relation est déjà en 3FN et la décomposition (sans perte d'information et qui préserve les dépendances) en schémas $(XA_1), \ldots, (XA_n)$ est superflue.*

Schéma 1 : La clé A incluse dans :

$$(AB, BC, AD, DC).$$

Les relations (AB),(AD) sont remplacées par (ABD). Donc la bonne décomposition est :

- (ABD) $A \rightarrow B$, $A \rightarrow D$
- (BC) $B \rightarrow C$
- (DC) $D \rightarrow C$

Schéma 2 : La clé unique EH incluse dans : EHS. Donc la bonne décomposition est :

- (CP) $C \rightarrow P$

1. On n'a pas besoin de vérifier si la décomposition est sans perte d'information et préserve les dépendances fonctionnelles.

- (HSC) $HS \rightarrow C$
- (HPS) $HP \rightarrow S$
- (CEN) $CE \rightarrow N$
- (EHS) $HE \rightarrow S$

Schéma 3 : La clé FN incluse dans : FNP. Donc la bonne décomposition est :

- (FA) $F \rightarrow A$
- (FNP) $FN \rightarrow P$

Schéma 4 : La clé MA incluse dans : MAD. Donc la bonne décomposition est :

- (MAD) $MA \rightarrow D$
- (MDR) $MD \rightarrow R$

Schéma 5 : La relation est déjà en 3FN (elle est même en BCNF), ce n'est pas la peine de la décomposer.

Schéma 6 : Idem

Schéma 7 : La clé est ACD. La relation n'est pas en 3FN. Le schéma (AB) est obtenu de la dépendance $A \rightarrow B$. On rajoute aussi le schéma (ACD) pour la clé. Donc la bonne décomposition est :

- (AB) $A \rightarrow B$
- (ACD) *(pas de dépendance)*

Schéma 8 : Les clés sont AC et BC. La relation n'est pas en 3FN. Le schéma (AB) est obtenu des dépendances $A \rightarrow B$ et $B \rightarrow A$. Le schéma (BD) est obtenu de la dépendance $B \rightarrow D$. On rajoute aussi le schéma (AC) pour la clé et on remplace (BD) par (ABD). Donc la bonne décomposition est :

- (ABD) $A \rightarrow B$, $B \rightarrow A$ *et* $B \rightarrow D$
- (AC) *(pas de dépendance)*

8.5 Forme Normale de Boyce-Codd

Exercice A : Soit le schéma

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{AB \rightarrow C, B \rightarrow D, C \rightarrow A\}$$

1. Donner une décomposition sans perte d'information et qui préserve les dépendances de $\mathcal{F}$.
2. Est-ce que cette décomposition est en BCNF?

Solution :

1. *Rappelons que toute relation a une décomposition (sans perte d'information et qui préserve les dépendances) en relations qui sont en 3FN. Mais ces relations ne sont pas forcement en BCNF.*
   Une relation en BCNF est une relation où les seules dépendances qui existent viennent des clés. Plus exactement, toute dépendance d'une relation en BCNF a pour membre gauche une superclé.
   On a trouvé qu'une bonne décomposition (sans perte d'information et qui préserve les dépendances de $\mathcal{F}$) du schéma $(ABCD)$ est :
   - (ABC) $AB \rightarrow C$ *et* $C \rightarrow A$.

- (BD) $B \rightarrow D$.

2. *Cependant les clés du schéma (ABC) sont AB et CB. Or dans la dépendance $C \rightarrow A$ de la relation (ABC), C n'est pas une superclé. Donc la décomposition $\Delta = (ABC,BD)$ n'est pas en BCNF.*

Exercice B : Parmi les schémas obtenus après la décomposition des schémas suivantes, trouver les décompositions en relations qui sont en BCNF.

Schéma 1 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B,B \rightarrow C,A \rightarrow D,D \rightarrow C\}$$

Schéma 2 :

$$\mathcal{R}(CPHSEN)$$

$$\mathcal{F} = \{C \rightarrow P,HS \rightarrow C,HP \rightarrow S,CE \rightarrow N,HE \rightarrow S\}$$

Schéma 3 :

$$\mathcal{R}(FANP)$$

$$\mathcal{F} = \{F \rightarrow A,FN \rightarrow P\}$$

Schéma 4 :

$$\mathcal{R}(MADR)$$

$$\mathcal{F} = \{MA \rightarrow D,MD \rightarrow R\}$$

Schéma 5 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{ABC \rightarrow D\}$$

Schéma 6 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B,A \rightarrow C,A \rightarrow D\}$$

Schéma 7 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B\}$$

Schéma 8 :

$$\mathcal{R}(ABCD)$$

$$\mathcal{F} = \{A \rightarrow B,B \rightarrow A,B \rightarrow D\}$$

Solution :

Toutes les relations obtenues après la décomposition des schéma données sont en BCNF. Prenons par exemple le schéma $\mathcal{R}(MADR)$.

La clé MA incluse dans : MAD. Donc la bonne décomposition est :

- (MAD) $MA \rightarrow D$
- (MDR) $MD \rightarrow R$

La relation (MAD) a pour clé MA qui est la partie gauche de la seule D.F. qui satisfait. De même pour la relation (MDR). Donc la décomposition obtenue est en BCNF.
Remarquons qu'une relation sans aucune contrainte est en BCNF[2] *(cf. exercices 7 et 8).*

Exercice C : Montre que le schéma

$$\mathcal{R}(\mathbf{R}ue,\mathbf{V}ille,\mathbf{C}odePostal)$$
$$\mathcal{F} = \{RV \rightarrow C, C \rightarrow V\}$$

n'est pas en BCNF.

Solution :
Les clés de cette relation sont RV et CR. Le membre gauche de la première dépendance est une clé (RV). Mais pas celui de la deuxième dépendance (C). Pourtant on a vu que comme le membre droit de la deuxième dépendance (V) appartient à une clé (RV), la relation est en 3FN.

2. Il faut noter qu'une relation sans aucune contrainte est toujours en toutes les formes normales.

Chapitre 9

Organisation Physique

Exercice A : On prend ici comme exemple la relation **Directeur** (*nom_directeur, nom_film*).

1. Organisation Séquentielle : Expliquer l'organisation séquentielle et représenter un exemple sur la relation Directeur. Montrer le fichier après une insertion et après quelques suppressions d'articles.

   Solution :

   Les articles sont stockés les uns derrière les autres dans des pages successives. L' avantage de cette structure physique est sa simplicité. En particulier, il n'existe pas d'ordre entre les articles. La seule façon d'accéder à un article est par balayage séquentiel*: on parcourt le fichier en séquence, page par page: chaque page est lue en mémoire: les articles dans la page sont alors parcourus séquentiellement. On lit chaque article et le sélectionne si la valeur de ses champs correspond au(x) critère(s) de recherche. Un exemple d'une organisation séquentielle pour la relation* **Directeur** *se trouve dans la Figure 9.1*

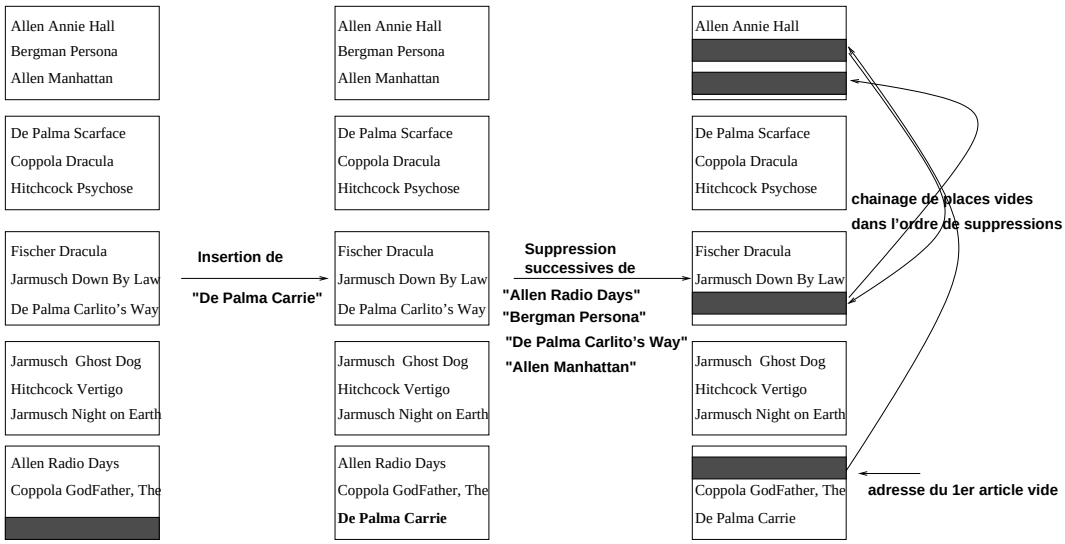


FIG. 9.1 – *Organisation Séquentielle de la relation* **Directeur**

2. Organisation Indexée : Montrer des exemples d'index non dense (primaire) et dense (secondaire) sur la relation Directeur.

   Solution :
   On illustre dans les Figures 9.2 et 9.3 des exemples d'index non dense respectivement sur l'attribut directeur et sur l'attribut nom de film. L'index est ordonné sur le même attribut que le fichier. Dans la figure 9.4 on montre deux index denses, l'un sur l'attribut directeur l'autre sur le nom du film. L'index est trié sur la clé alors que l'ordre des articles dans le fichier est quelconque. Deux remarques sont importantes: on a supposé dans ces exemples que l'index tient dans une page (feuille unique). Il faut se rappeler qu'en pratique, l'index a une structure arborescente (Arbre B, voir http://deptinfo.cnam.fr/CycleA/SD/ pour une démonstration en ligne de l'arbre B et des tris). Ensuite, l'exemple dans la figure 9.4 suppose qu'on associe à une valeur de clé "c" dans l'index une adresse de page (où se trouvent un ou plusieurs nuplets ayant "c" pour valeur de l'attribut clé). En fait la plupart du temps, ce n'est pas une adresse de page, mais une adresse complète du nuplet (adresse de page et adresse dans la page, voir cours) qui est associée à "c" dans l'index. Ceci a pour conséquence, que s'il y a "n" nuplets dans la page ayant pour valeur de clé "c", il y aura "n" couples ("c, adresse de nuplet) dans l'index.

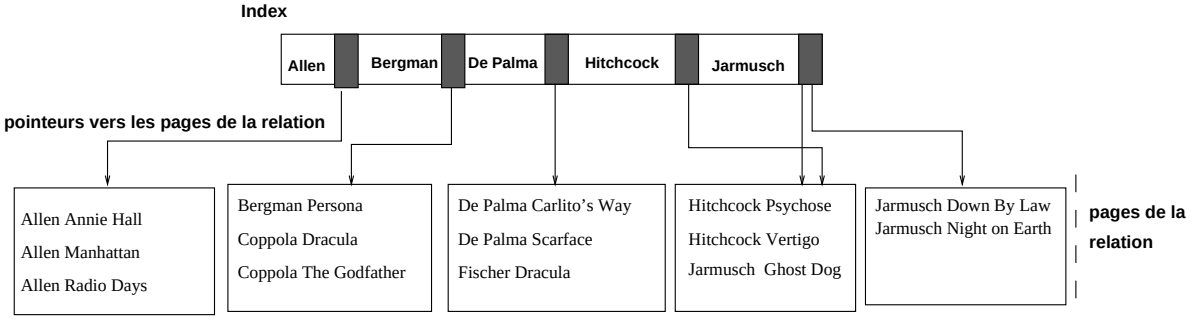


FIG. 9.2 – *Index non dense sur l'attribut* nom_directeur *de la relation* **Directeur**

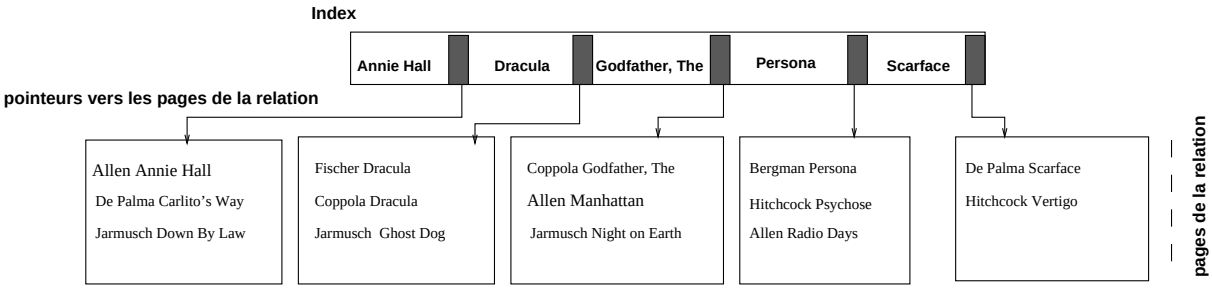


FIG. 9.3 – *Index non dense sur l'attribut* nom_film *de la relation* **Directeur**

Exercice B : Construire un index sur la date de naissance des musicien (arbre B, ordre 2) :

```
Monteverdi    1589
Couperin      1668
Bach          1685
Rameau        1684
Debussy       1862
Ravel         1875
Mozart        1756
```

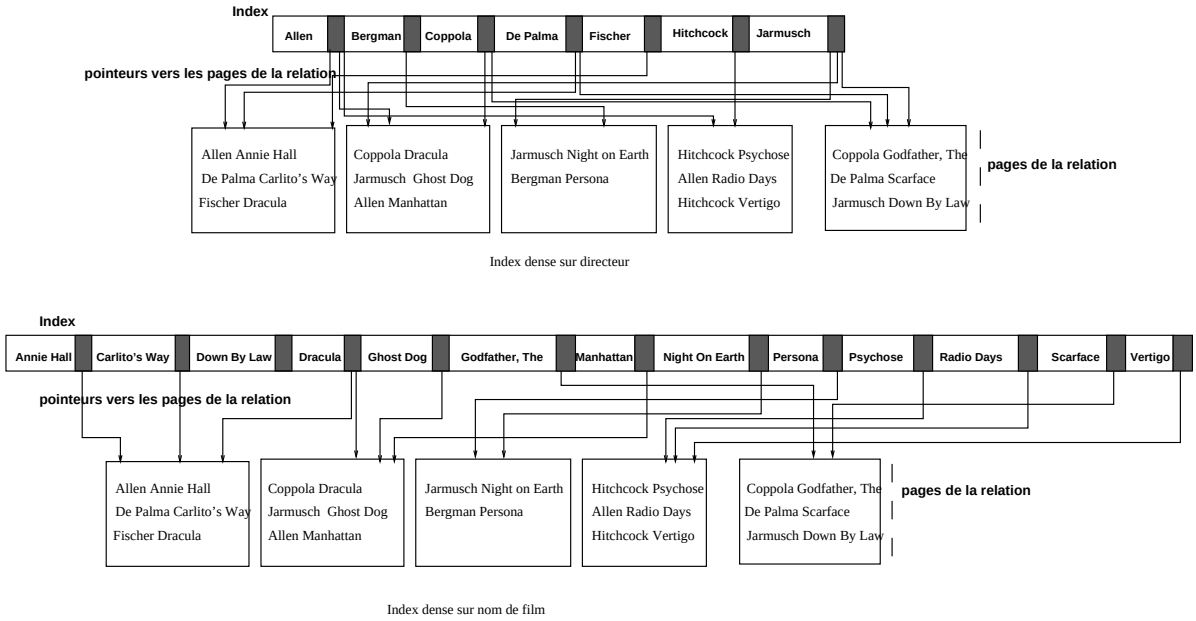
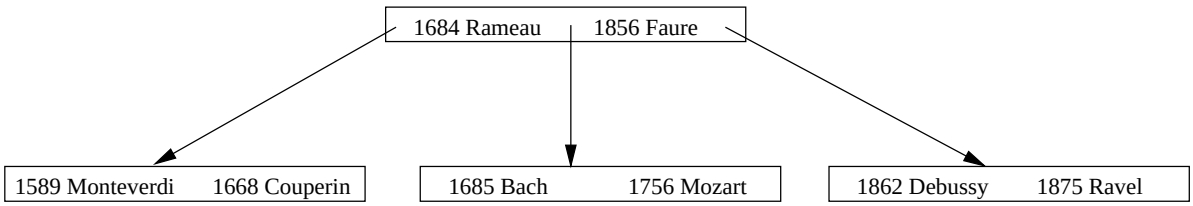


FIG. 9.4 – *Index dense*

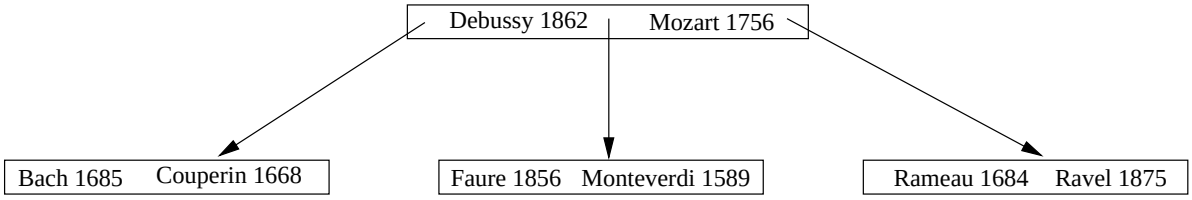
```
Faure          1856
```

Solution :



Exercice C : Construire un index sur les noms des musicien (arbre B, ordre 2).

Solution :

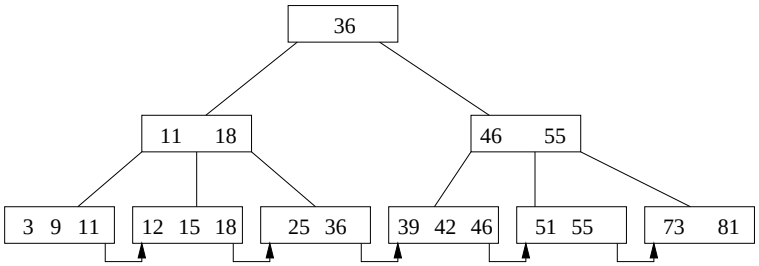


Exercice D : Construire un arbre B+ d'ordre 2 sur les numéros de département.

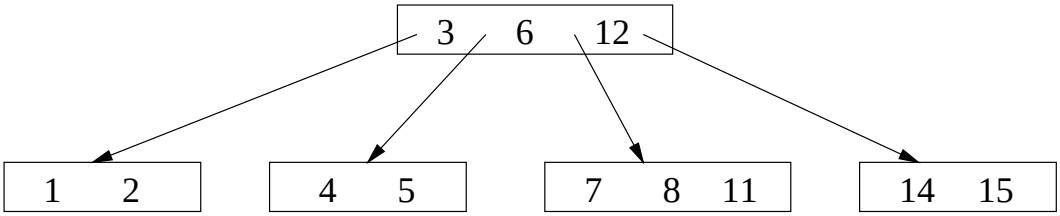
```
3       Allier
36      Indre
18      Cher
9       Ariège
11      Aude
12      Aveyron
73      Savoie
55      Meuse
46      Lot
39      Jura
```

```
81      Tarn
25      Doubs
15      Cantal
51      Marne
42      Loire
```

Solution :

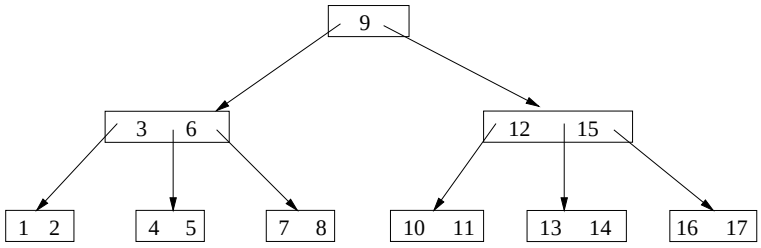


Exercice E : Soit le fichier séquentiel suivant (on ne donne pour chaque article du fichier que la clé sur laquelle on construit l'arbre): 1 15 3 12 6 4 11 7 2 5 14 8 9 17 10 13 16. L'index en arbre B d'ordre 2 après l'insertion des clés 1 15 3 12 6 4 11 7 2 5 14 8 est montré dans la figure suivante :



1. Donnez l'arbre résultant après l'insertion de tous les articles du fichier séquentiel.

   Solution :



2. Combien de nœuds différents (racine et feuilles compris) doit-on parcourir dans l'index pour répondre à la requête qui cherche les articles dont la clé appartient à l'intervalle [5,10].

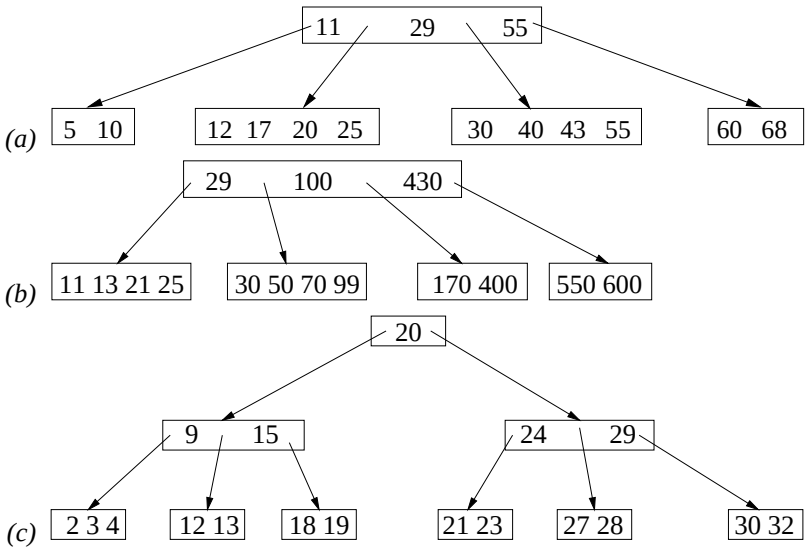
   Solution :
   Il faut parcourir 6 noeuds : [9], [3,6], [4,5], [7,8] [12, 15], [10,11].

Exercice F : Soit les fichiers séquentiels suivants (on ne donne pour chaque article du fichier que la clé sur laquelle on construit l'arbre) :

- 5, 29, 17, 68, 60, 43, 10, 11, 12, 20, 55, 30, 40, 50, 25.
- 100, 29, 170, 70, 600, 430, 99, 11, 13, 21, 550, 30, 400, 50, 25
- 2 15 30 28 12 4 18 19 24 29 13 27 9 20 3 32 21 23

1. Construire un index en arbre B d'ordre 2 pour chacun des fichier.

   Solution :



2. Construire un index en arbre B+ d'ordre 2 pour chacun des fichier.

Chapitre 10

Algorithmes de Jointure

Soit les relations suivantes:

- **Directeur**(*nom_directeur, nom_film*)
- **Acteur**(*nom_film, nom_acteur*)

Soit la requête suivante :

```
SELECT nom_directeur, nom_acteur
FROM Directeur Join Acteur
WHERE Directeur.nom_film = Acteur.nom_film
```

Pour évaluer cette requête, il faut calculer la jointure **Directeur** $\bowtie$ **Acteur**. Pour l'éxecution des jointures, décrivez et évaluez la complxité de l'algorithme avec boucles imbriquées et avec tri-fusion. Dans les deux cas on tiendra compte des mouvements entre mémoire centrale et disque et on évluera le nombre d'entrées-sorties.

Exercice A : *Algorithme avec boucles imbriquées*;

Solution :

Algorithme

On suppose que les relations **Directeur** *et* **Acteur** *sont stockées* séquentiellement*: elles ne possèdent aucun chemin d'accès privilégié (aucun index) (Figure 10.1). L'algorithme le plus simple pour évaluer la jointure* **Directeur** $\bowtie$ **Acteur** *consiste à comparer chaque nuplet de la relation* **Directeur** *à* tous *les nuplets de la relation* **Acteur**, *puis a concaténer les nuplets pour lesquels* **Directeur**.nom_film = **Acteur**.nom_film *et enfin à ajouter le nouveau tuple au résultat. L' algorithme est esquissé dans le Tableau 10.1* [1]*:*
Pour évaluer le coût (le nombre des E/S) de cet algorithme il faut connaitre les paramètres suivants: (a) la taille de chacune de deux relations et (b) la taille disponible en Mémoire Centrale pour réaliser cette opération.
Supposons que les deux relations à joindre ont une taille supérieure à la taille de l'espace Mémoire Centrale disponible. On suppose un tampon en m'éoire centrale de $b + 2$ pages. On affecte en Mémoire centrale b pages (tampon principal) pour lire des nuplets de la table **Directeur**, *une page (tampon auxiliaire) pour lire des nuplets de la table* **Acteur** *et une page pour stocker le résultat de la jointure (tampon écriture). On charge* séquentiellement *la relation* **Directeur** *par paquets de b pages dans le tampon principal (Figure 10.1).*

1. Pour l'algorithme on joint les relations R, S où l'attribut de jointure est a.

R$\acute{e}sultat$ = Ø
tant que R n'est pas finie
    lire tuple t_R de R
        tant que S n'est pas finie
            lire tuple t_S de S
                si $t_R.a = t_S.a$, Résultat = Résultat ∪ { $t_R \bowtie t_S$ }
        ftq
    ftq

TAB. 10.1 – *Esquisse de l'algorithme des Boucles Imbriquées*

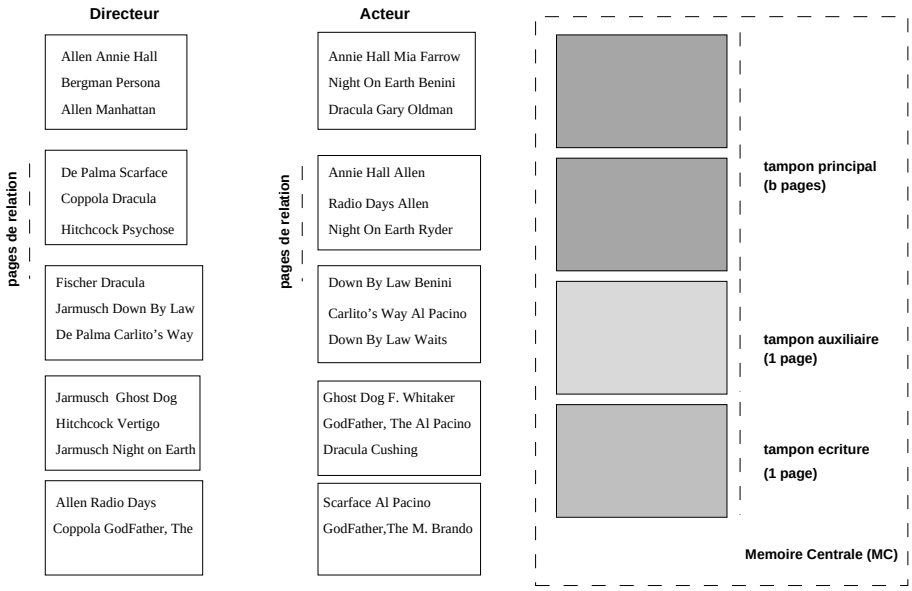


FIG. 10.1 – *Relations* **Directeur** *et* **Acteur** *sur disque, et image de la Mémoire Centrale*

Quand le tampon principal est plein, on charge la première page de la relation **Acteur** *dans le tampon auxiliaire. Ensuite, on compare chaque nuplet du tampon principal à tous les nuplets du tampon auxiliaire. Les nuplets qui vérifient le critère de jointure sont joints. Le nuplet résultat est alors écrit dans le tampon écriture. Lorsque le tampon écriture est plein, son contenu est écrit sur disque.*
Une fois que la jointure des b premières pages de **Directeur** *est effectuée avec la première page de* **Acteur**, *on réitère le traitement en chargeant la seconde page de* **Acteur** *dans le tampon auxiliaire et ainsi de suite jusqu'à épuisement de la relation* **Acteur**. *A ce moment là, tous les nuplets des b premières pages de la relation* **Directeur** *chargées en MC, ont été comparés avec tous les nuplets de la relation* **Acteur**. *On réitère alors le traitement en chargeant les b pages suivantes de la relation* **Directeur** *dans le tampon principal, et ce jusqu'à épuisement de la relation* **Directeur**.

Performances

La relation **Directeur** *est lue une seule fois ($p_{Directeur}$ E/S)*[2] *par morceaux de b pages. Pour chaque morceau, la relation* **Acteur** *est lue entièrement. Alors, la relation* **Directeur** *doit être découpée en $\lceil p_{Directeur}/b \rceil$ morceaux (où $\lceil A \rceil$ dénote la partie entière supérieure de A). Le nombre total de*

2. Ici p_R dénote le nombre de pages de la relation R, et n_R dénote le nombre de nuplets de la relation R.

lectures générées par l'algorithme de jointure par boucles imbriquées est:

$$Coût E/S(Join\_Boucles\_Imbr) = p_{Directeur} + \lceil p_{Directeur}/b \rceil \times p_{Acteur}$$

Le nombre d'écritures dépend de la taille du résultat. Compte tenu de la formule du coût d'E/S on a intérêt à charger la plus petite relation dans le tampon principal et à faire défiler la plus grande relation dans le tampon auxiliaire. Dans le cas où la plus petite relation tient en MC, alors le coût d'E/S devient $p_{Directeur} + p_{Acteur}$.
Les nombre de comparaisons en mémoire centrale est $n_{Directeur} \times n_{Acteur}$.

Exercice B : *Algorithme avec tri-fusion*;

Solution :
La première étape consiste à trier chacune des relations **Directeur** *et* **Acteur** *sur leur attribut de jointure (*nom_film*) (Figure 10.2). La deuxième étape (fusion) nécessite pour son exécution trois tampons du Mémoire Centrale, d'une page chacun. Les deux premiérs tampons permettent de lire séquentiellement les relations* **Directeur** *et* **Acteur** *page par page et le troisième tampon est réservé pour écrire le résultat de la jointure.*

Algorithme

L'algorithme consiste à positionner un pointeur courant $pt_{Directeur}$ *(*pt_{Acteur}*) sur le premiér nuplet de la relation* **Directeur** *(***Acteur***) et à comparer les attributs de jointure de ces deux nuplets. L'algorithme répète le test suivant:*

- *Si l'attribut* **Directeur**.nom *du nuplet pointé par* $pt_{Directeur}$ *est inférieur à l'attribut* **Acteur**.nom *du nuplet pointé par* pt_{Acteur}*, on incrémente* $pt_{Directeur}$ *afin qu'il pointe sur le nuplet suivant de* **Directeur**.
- *Si l'attribut* **Directeur**.nom *du nuplet pointé par* $pt_{Directeur}$ *est supérieur à l'attribut* **Acteur**.nom *du nuplet pointé par* pt_{Acteur}*, on incrémente* pt_{Acteur} *afin qu'il pointe sur le nuplet suivant de* **Acteur**.
- *Si les deux nuplets courants ont leurs attributs de jointure égaux, on génère un nuplet résultat et on incrémente l'un des deux pointeurs.*

Lorsque l'un des pointeurs courants pointe sur le dernier nuplet d'une page, son incrémentation génère la lecture de la page suivante de la relation, dans le tampon associé à cette relation, et le pointeur est positionné sur le premiér nuplet de cette nouvelle page.
Une variante de l'algorithme ci-dessus peut etre utilisée pour les theta-jointures *autres que l'*equi-jointure.

Performances

Le nombre d' E/S de la jointure par tri-fusion comprend le coût éventuel du tri de la (ou des) relations à joindre[3] *plus le coût de lecture séquentielle des deux relations à joindre, plus le coût d'écriture sur disque du résultat (ce dernier est exclus de la formule ci-dessous)*

$$cout(Join\_tri\_fusion) = \begin{cases} p_S + p_R, si\ R, S\ sont\ deja\ triees, \\ 2p_R \times log_b p_R + p_S + p_R, si\ S\ est\ triée \\ 2p_S \times log_b p_S + p_S + p_R, si\ R\ est\ triée \\ 2p_R \times log_b p_R + 2p_S \times log_b p_S + p_S + p_R, si\ ni\ R, ni\ S\ ne\ sont\ triées \end{cases}$$

Le coût mémoire centrale de fusion de deux listes triées de n_R *et* n_S *nuplets dans le pire des cas est quadratique. Dans un cas non dégénéré il est linéaire en* $n_R + n_S$*. Noter aussi que dans le cas dégénéré, l'algorithme montré plus haut ne marche pas. Pourquoi?*

3. Le coût du tri d'une relation de p pages est de l'ordre de $2p \times log_b p$ E/S b étant le nombre de pages du tampon en mémoire centrale disponible pour le tri.

FIG. 10.2 – *Relations* **Directeur** *et* **Acteur** *triées sur* nom_film

Chapitre 11

Optimisation de Requêtes

Exercice A : Soit la base STATION DE SKI de schéma:

```
hotel (noms,nomh,categorie,adresse,tel,nb_chambres)
station (noms,gare)
activite (type_activite,noms)
```

Pour chacune des requêtes suivantes, on demande:

1. l'arbre syntaxique de la requête
2. le plan d'exécution obtenu par la restructuration algébrique
3. le plan d'exécution obtenu par une optimisation globale.

Requête 1 : adresse, numéro de téléphone et nombre de chambres des hôtels de catégorie 3' dans la station de nom (*noms* 'persey'.

```
SELECT adresse, tel, nb_chambres
FROM hotel
WHERE noms='pesey' AND categorie=3;
```

Solution :

(1) arbre syntaxique de la requête : *Figure 11.1*

(2) restructuration algébrique : *Les principes qu'il faut prendre en compte pour la restructuration algébrique sont les suivants:*

1. *évaluer les sélections (ou restrictions) le plus tôt possible. En effet, la relation qu'on obtient par l'évaluation de sélections est plus petite que la relation initiale.*
2. *faire des projections pour réduire la taille de la relation en question.*
3. *permuter les jointures. En principe* $(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$ *si et seulement si, l'attribut commun est le même pour les trois relations.*

Pour la requête ci-dessus, aucune restructuration algébrique n'est nécessaire.

(3) optimisation globale : *Pour trouver le plan d'exécution "optimal", il faut estimer le coût d'E/S pour chaque opération algébrique.*
*Pour évaluer la sélection (*noms='pesey' AND catégorie=3*), plusieurs stratégies existent: utiliser les index existants de la relation "hotel" ou balayage séquentiel de cette relation. L'utilisation de l'index permet d'avoir un accès direct aux tuples de la relation qui satisfont le prédicat de sélection.*

balayage séquentiel : *on parcourt séquentiellement la relation: on lit en MC chaque page de la relation en séquence, on lit en séquence les tuples de la page courante on*

évalue le prédicat en Mémoire Centrale (MC). Le coût en E/S de l'évaluation de la sélection est le coût de lecture de la relation en MC (dans la suite de l'exercice on ne compte pas le cout d'écriture du résultat sur le disque qui est le même quelle que soit la stratégie.

$$coût(selection) = \ nombre\ pages\ de\ la\ relation\ hotel \ = \ 1000$$

.

index non dense sur noms : *On utilise l'index pour trouver et lire en MC la page qui contient les tuples qui satisfont la contrainte (*noms='pesey'*). Le pr'edicat (les contraintes* noms = 'pesey' *et* catégorie=3 *sont vérifiées sur tous les tuples de la page. Le coût de la sélection est estimé par la formule:*

$$Sélectivité\_prédicat\_de\_la\_clé \times (NPI(hotel) + NPR(hotel))(1)$$

- *Sélectivité_prédicat_de_la_clé (*noms*);*
- $NPI(hotel)$ *est le nombre des pages feuilles de l'index plaçant de la relation* **hôtel***;*
- $NPR(hotel)$ *est le nombre de pages de la relation* **hôtel**.

*La sélectivité du prédicat (*noms='pesey'*) est calcule par la formule:*

$$Sélectivité\_prédicat\_sur\_clé = 1/(nb\ valeurs\ diffférentes; de\ la\ cle\ (noms))$$

Pour la requête donnée, on n'a pas directement l'information sur le nombre de valeurs différentes de noms*, mais on peut l'obtenir à partir de la relation* **station**. *En effet* noms *est le clé primaire de la relation* **station**, *et le nombre de valeurs différentes est 1000 (le nombre de tuples de la relation* **station***). Alors comme on a une contrainte référentielle pour l'attribut* noms *de la relation* **hôtel**, *le nombre de valeurs différentes de ce attribut pour hôtel est au maximum 1000. Donc, la sélectivité du prédicat sur* noms *est 1/1000 = 0.001*
A partir de la formule (1), le coût d'E/S est 0.001 × (26 + 1000) = 1.026 E/S (accès direct).

index dense sur catégorie: *Le coût de l'évaluation du prédicat dans ce cas est calculé par la formule:*

$$Sélectivité\_prédicat\_de\_la\_clé \times (NPI(hotel) + NPT(hotel))$$

- *Sélectivité_prédicat_de_la_clé (*catégorie*);*
- *NPI est le nombre des pages feuilles de l'index non plaçant de la relation* **hôtel***;*
- *NPT(R) est le nombre de tuples de la relation* **hôtel**.

*La sélectivité du prédicat (*catégorie=3'*) est estimée par la formule:*

$$Sélectivité\_prédicat\_sur\_clé = 1/(nb\ valeurs\ diffférentes\ de\ la\ cle\ (categorie))$$

Pour la requête donnée, la sélectivité du prédicat sur l'attribut catégorie *est 1/5. Donc, le coût d'évaluation de la sélection est : 1/5 × (33 + 10000) = 2007 E/S. Le coût est trop élevé (l'index n'a aucun intérêt car le nombre de catégories différentes est trop faible).*

Requête 2 : nom de station (noms) et la gare de la station pour le sstations ayant pour activité le tennis.

```
SELECT noms, gare
FROM station, activite
WHERE type_activite = ''tennis''
AND station.noms=activite.noms
```

Solution :
arbre syntaxique de la requête: *Figure 11.1*

restructuration algébrique : *Figure 11.1*

optimisation globale : *Pour choisir le plan d'exécution le meilleur, il faut estimer le coût d'E/S minimum pour chacun des opérations algébriques et choisir le moins cher. Le coût de cette requête est:*

$$coût(requete) = coût(évaluation\_prédicat\_réstriction) + coût(évaluation\_jointure)$$

- coût(évaluation_prédicat_restriction)*:*

  balayage séquentiel : *Le coût d'évaluation de la restriction est égal au nombre pages de la relation* **activité** *= 300.*

  index dense sur type_activité : *Le coût d'E/S est estimé par:*

  $$Sélectivité\_prédicat\_de\_la\_clé \times (NPI(hotel) + NPT(hotel))$$

  Pour la requête donnée, la sélectivité du prédicat sur l'attribut type_activité *est 1/(nb différentes de valeurs de* type_activité*) = 1/40. Alors, le coût d'évaluation de la sélection est : 1/40 × (10 + 3000) = 76 E/S. La relation obtenue n'est pas dans l'ordre de noms.*
- coût(évaluation_jointure)*: Pour l'évaluation de jointure il existe plusieurs algorithmes qu'on peut utiliser: (a) boucles imbriquées, (b) tri-fusion. Pour calculer le coût de la jointure il faut d'abord calculer la taille de la relation* **Temp** *(Figure 11.1). La taille d'une relation après avoir évalué le prédicat de la sélection est donné par la formule :*

  $$Séléctivite\ du\ prédicat\ réstriction\ \times\ NPT(R)$$

  La sélectivité du prédicat **type_activité = 'tennis'** *est 1/40, alors le nombre des tuples lus en MC (taille de la relation* **Temp***) est 1/40 × 3000 = 75 tuples. Comme la relation* **Temp** *est constituée d'un seul attribut (*noms*), alors on peut considérer qu' elle tient dans 1 ou au plus 2 pages.*

  boucles imbriquées : *Le coût de l'évaluation de la jointure est :*

  $$coût(station \bowtie Temp) = \begin{cases} NBP_{station} + \lceil NBP_{station}/b \rceil \times NBP_{Temp}, \\ si\ \textbf{station}\ est\ la\ petite\ relation \\ NBP_{Temp} + \lceil NBP_{Temp}/b \rceil \times NBP_{station}, \\ si\ \textbf{Temp}\ est\ la\ petite\ relation \\ NBP_{Temp} + NBP_{station}, \\ si\ une\ de\ deux\ relations\ tient\ en\ MC \end{cases}$$

  Dans notre cas, la relation **Temp** *tient en MC (2 pages au maximum) alors le coût(*$station \rtimes Temp$*) =* $NBP_{Temp} + NBP_{station}$ *= 2 + 100 = 102 E/S. Dans le cas où* **Temp** *est déjà en MC (après l'évaluation de la sélection et de la projection), le coût(*$station \rtimes Temp$*) =* $NBP_station$ *= 100 E/S.*

  tri-fusion : *Le coût d'évaluation de la jointure est :*

  $$coût(station \rtimes Temp) = NBP_{station} + NBP_{Temp} + coût(tri\_station) + coût(tri\_Temp)$$

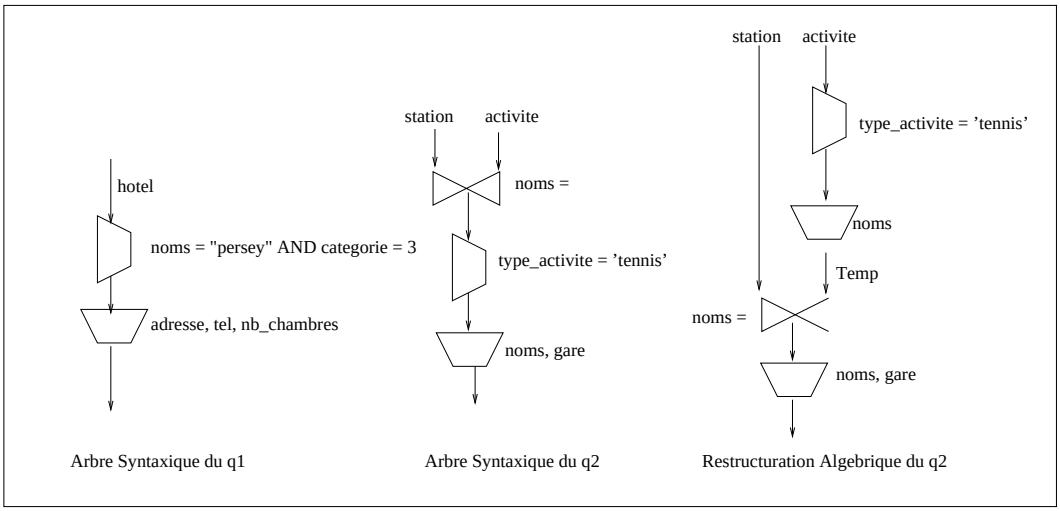


FIG. 11.1 – *Arbres Syntaxique et Restructuration Algébrique pour les requêtes (1) et (2)*

Exercice B : Soit le schéma suivant:

```
CREATE TABLE FILM (                  CREATE TABLE VU (
       TITRE   VARCHAR2(32),                SPECTATEUR      VARCHAR2(32),
       REALISATEUR VARCHAR2(32),            TITRE           VARCHAR2(32)
       ACTEUR VARCHAR2(32)                );
);
```

Soit la requête SQL:

```
SELECT ACTEUR, REALISATEUR
  FROM FILM, VU
 WHERE FILM.TITRE=VU.TITRE
```

Dans chacun des cas suivants, donner l'algorithme de jointure de ORACLE (par EXPLAIN, un arbre d'exécution commenté, une explication textuelle ou tout autre moyen de votre choix):

1. Il n'existe pas d'index sur TITRE ni dans FILM ni dans VU,

   Solution :
   Tri-fusion (voir cours): Figure 11.2 et plan d'exécution Oracle.

```
Plan d'exécution
----------------------------------------------------------------------
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 TABLE ACCESS FULL VU
    4 SORT JOIN
      5 TABLE ACCESS FULL FILM
```

2. Il existe un index sur TITRE dans FILM seulement.

   Solution :
   La table VU est prise comme table directrice: pour chaque titre dans VU parcouru séquentiellement, on cherche à travers l'index le ROWID s'il existe d'un nuplet de FILM et on fait la jointure (comme la relation FILM n'est pas normalisée, il y a plusieurs nuplets de même titre) (Figure 11.3).

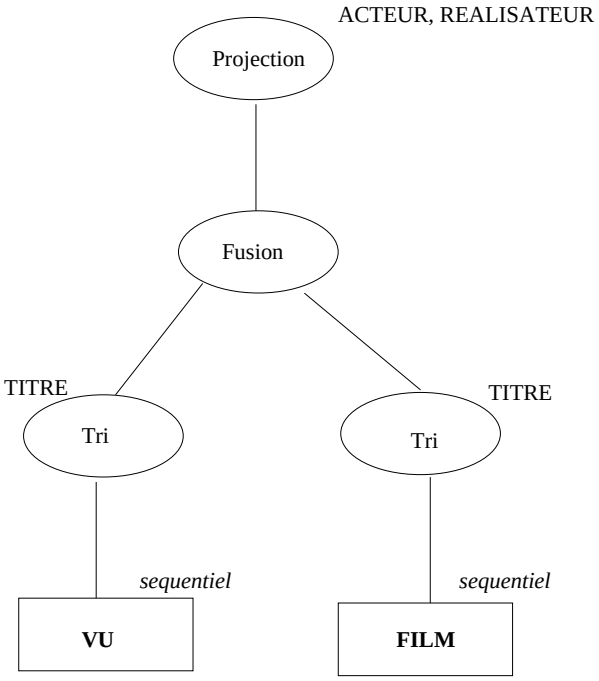


FIG. 11.2 – *Plan d'exécution*

```
CREATE INDEX FILM_TITRE_idx on FILM (TITRE);

Plan d'exécution
----------------------------------------------------------------------
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS FULL VU
    3 TABLE ACCESS BY ROWID FILM
      4 INDEX RANGE SCAN FILM_TITRE_IDX
```

3. Il existe un index sur TITRE dans les deux relations.

   Solution :
   idem (l'optimiseur choisit la dernière table comme table directrice de la clause FROM quand il existe un index sur la colonne de jointure dans les deux tables)

```
CREATE INDEX FILM_TITRE_idx on FILM (TITRE);
CREATE INDEX VU_TITRE_idx on VU (TITRE);

Plan d'exécution
----------------------------------------------------------------------
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS FULL VU
    3 TABLE ACCESS BY ROWID FILM
      4 INDEX RANGE SCAN FILM_TITRE_IDX
```

Exercice C : Soit la requête :

```
SELECT  e.enom, d.dnom
FROM    emp e, dept d
```

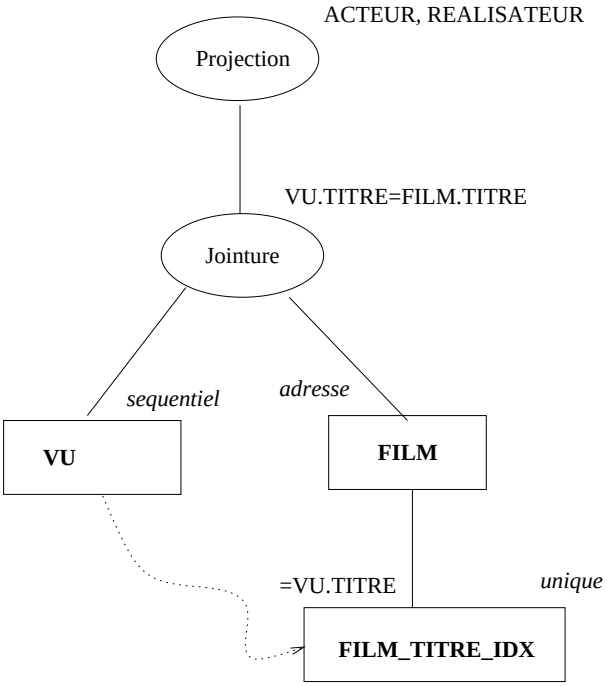


FIG. 11.3 – *Plan d'exécution*

```
WHERE   e.dno = d.dno
AND     e.sal = 10000
```

sur la relation **EMP** de schéma *(EMPNO, SAL, MGR, DNO)*. Cette requête affiche le nom des employés dont le salaire (SAL) est égal à 10000, et celui de leur département. Indiquez le plan d'exécution dans chacune des hypothèses suivantes.

1. Index sur $DEPT(Dno)$ et sur $EMP(Sal)$

   Solution :
   Figure 11.4.

```
Plan d'execution
----------------------------------------------------------------------------
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS BY ROWID EMP
      3 INDEX RANGE SCAN EMP_SAL
    4 TABLE ACCESS BY ROWID DEPT
      5 INDEX UNIQUE SCAN DEPT_DNO
```

   Boucles imbriquées (`NESTED LOOP`*) : on choisit de parcourir un sous-ensemble de EMP en utilisant l'index, puis on accède a DEPT avec l'index sur DEPTNO.*

2. Index sur $EMP(Sal)$ seulement.

   Solution :
   Figure 11.5.

```
Plan d'execution
----------------------------------------------------------------------------
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS FULL DEPT
    3 TABLE ACCESS BY ROWID EMP
```

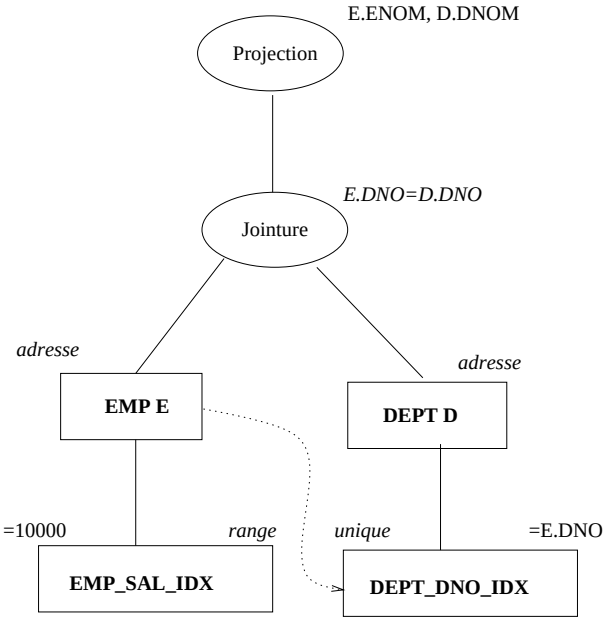


FIG. 11.4 – *Plan d'exécution*

```
      4 INDEX RANGE SCAN EMP_SAL
```

Algorithme de SCAN-INDEX. Equivalent à une jointure par NESTED-LOOP brutal. On pourrait (i) changer l'ordre des tables sans modifier la complexité, et (ii) faire un tri-fusion.

3. Index sur $EMP(Dno)$ et sur $EMP(Sal)$

   Solution :
   Figure 11.6.

```
Plan d'execution
----------------------------------------------------------------------------
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 TABLE ACCESS FULL DEPT
    3 TABLE ACCESS BY ROWID EMP
      4 AND-EQUAL
        5 INDEX RANGE SCAN EMP_DNO
        6 INDEX RANGE SCAN EMP_SAL
```

   Comme l'index sur EMP(DNO) n'est pas unique, on a intérêt à limiter la liste des adresses de tuples en utilisant les deux index et en faisant l'intersection.

4. Voici une autre requête, légèrement différente. Plan d'exécution s'il n'y a pas d'index.

```
SELECT e.enom
FROM emp e, dept d
WHERE  e.dno = d.dno
AND    d.ville = 'Paris'
```

   Solution :

```
Plan d'execution
----------------------------------------------------------------------------
0 SELECT STATEMENT
```

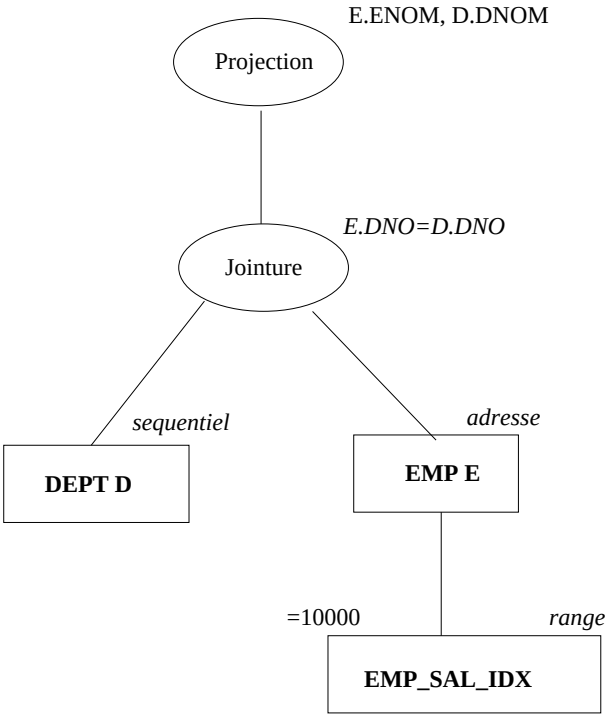


FIG. 11.5 – *Plan d'exécution*

```
1 MERGE JOIN
  2 SORT JOIN
    3 TABLE ACCESS FULL DEPT
  4 SORT JOIN
    5 TABLE ACCESS FULL EMP
```

Algorithme de tri-fusion classique.

5. Que pensez-vous de la requête suivante par rapport à la précédente ?

```
SELECT e.enom
FROM emp e
WHERE  e.dno IN (SELECT d.dno
                 FROM Dept d
                 WHERE d.Ville = 'Paris')
```

Voici le plan d'exécution donné par ORACLE :

```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 TABLE ACCESS FULL EMP
    4 SORT JOIN
      5 VIEW
        6 SORT UNIQUE
          7 TABLE ACCESS FULL DEPT
```

Qu'en dites vous ?

Solution :

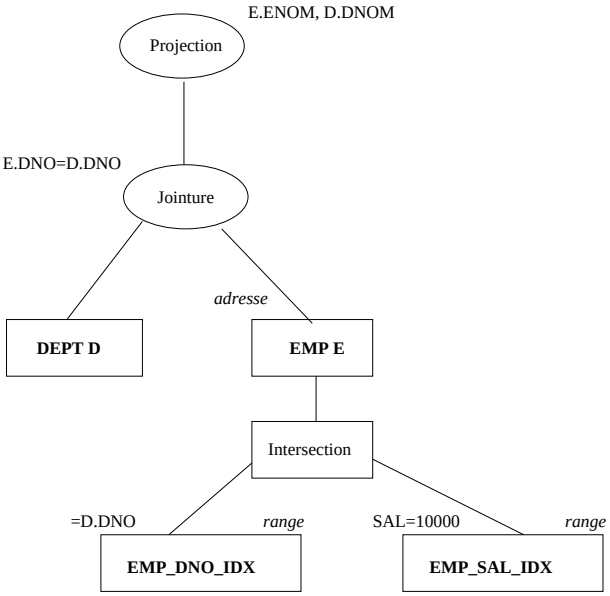


FIG. 11.6 – *Plan d'exécution*

Création d'une table temporaire (`VIEW`*) contenant les numéros des départements à Paris. On a éliminé les doublons (*`SORT UNIQUE`*). Ensuite on fait un tri-fusion. Donc exécution différente pour une requête équivalente.*

Exercice D : Sur le même schéma, voici maintenant la requête suivante.

```
SELECT *
FROM EMP X WHERE X.SAL IN (SELECT SAL
                             FROM EMP
                            WHERE EMP.EMPNO=X.MGR)
```

Cette requête cherche les employés dont le salaire (SAL) est égal à celui de leur patron (MGR). On donne le plan d'exécution avec Oracle (outil EXPLAIN) pour cette requête dans deux cas: (i) pas d'index, (ii) un index sur le salaire et un index sur le numéro d'employé. Expliquez dans les deux cas ce plan d'exécution (éventuellement en vous aidant d'une représentation arborescente de ce plan d'exécution).

1. Pas d'index.

```
Plan d'exécution
----------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
  2 TABLE ACCESS FULL EMP
```

   Solution :
   Figure 11.7. Boucles imbriquées (FILTER) : On parcourt la table EMP (ligne 2); pour chaque employé, on regarde le salaire SAL et le numéro du patron MGR; on parcourt alors la table EMP (ligne 3) pour trouver l'employé dont le numéro est MGR et on compare le salaire à SAL. Donc c'est une boucle imbriquée brutale : on aurait pu faire un tri-fusion.

2. Index empno et index sal.

```
Plan d'exécution
----------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
```

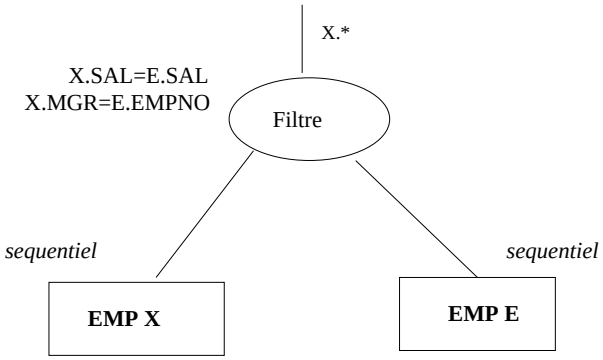


FIG. 11.7 – *Plan d'exécution*

```
2 AND-EQUAL
  3 INDEX RANGE SCAN I-EMPNO
  4 INDEX RANGE SCAN I-SAL
```

Solution :

Figure 11.8. Boucles imbriquées (FILTER) : on parcourt la table EMP (ligne 2). Pour chaque

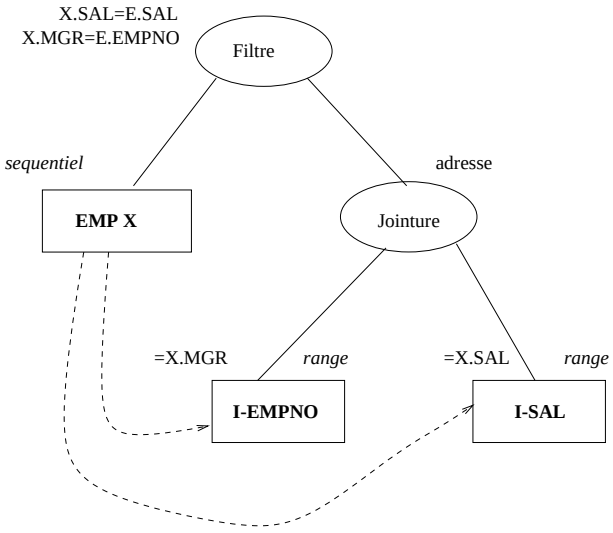


FIG. 11.8 – *Plan d'exécution*

employé, le salaire SAL et le numéro $EMPNO$, valeur de l'attribut MGR servent de clé pour accéder à l'index (lignes 4 et 5). L'intersection des listes de row-id (ligne 3) obtenues par les étapes 4 et 5 donne si elle est non vide un row-id de patron ayant même salaire que l'employé

3. Dans le cas où il y a les deux index (salaire et numéro d'employé) et où la requête est :

```
SELECT *
  FROM EMP X
 WHERE X.SAL = (SELECT SAL
                  FROM EMP
                 WHERE EMP.EMPNO=X.MGR)
```

on a le plan d'exécution suivant:

```
Plan d'exécution
```

```
--------------------------------------------------------------------------
0 FILTER
  1 TABLE ACCESS FULL EMP
  2 TABLE ACCESS ROWID EMP
    3 INDEX RANGE SCAN I-EMPNO
```

Expliquez-le.

Solution :

Dans ce cas, seul l'index sur les numéros d'employés est utilisé. Boucles imbriquées (FILTER); on parcourt la table EMP: pour chaque employé, l'attribut MGR donne le numéro d'employé de son patron. On accède à son patron par l'index sur les numéros d'employé (lignes 4 puis 3): on vérifie alors si son salaire est égal à celui de l'employé.

Exercice E : On reprend le schéma CINEMA donné dans le cours, **mais on ne sais plus quels index existent**.

Questions:

1. Donner l'ordre SQL pour la requête: *Quels sont les films d'Hitchcock visibles après 20h00 ?*

   Solution :

   ```
   SELECT F.TITRE
     FROM SEANCE S, FILM F, ARTISTE A
    WHERE A.NOM = 'Hitchcock'
      AND F.ID-REALISATEUR = A.ID-REALISATEUR
      AND S.ID-FILM = F.ID-FILM
      AND S.HEURE_DEBUT > '20:00'
   ```

2. Donner l'expression algébrique correspondante et proposez un arbre de requête qui vous paraît optimal.

   Solution :

   $\sigma_{HEURE\_DEBUT>'20:00'} SEANCE \bowtie (\sigma_{NOM='Hitchcock'} ARTISTE \bowtie_{ID-Artiste=ID-Realisateur} FILM)$

3. Sous ORACLE, l'outil EXPLAIN donne le plan d'exécution suivant:

   ```
   0 SELECT STATEMENT
     1 MERGE JOIN
       2 SORT JOIN
         3 NESTED LOOPS
           4 TABLE ACCESS FULL ARTISTE
           5 TABLE ACCESS BY ROWID FILM
             6 INDEX RANGE SCAN IDX-ARTISTE-ID
       7 SORT JOIN
        8 TABLE ACCESS FULL SEANCE
   ```

   Commentez le plan donné par EXPLAIN. Pourrait-on améliorer les performances de cette requête?

   Solution :

   L'existence d'un tri-fusion indique qu'il manque un index. Ici on peut résoudre le pb en créant un index sur $SEANCE(id - Film)$.

Exercice F : Soit le schéma suivant:

```
CREATE TABLE Artiste (                CREATE TABLE Film (
       ID-artiste     NUMBER(4),             ID-film        NUMBER(4),
       Nom            VARCHAR2(32),          Titre          VARCHAR2(32),
```

```
      Adresse       VARCHAR2(32)          Année          NUMBER(4),
);                                        ID-réalisateur NUMBER(4)
                                      );

CREATE TABLE Joue (
      ID-artiste    NUMBER(4),
      ID-film       NUMBER(4)
);
```

Questions:

1. Donner l'ordre SQL pour la requête: *Afficher le nom des acteurs et le titre des films où ils ont joué.*
2. Donner l'expression algébrique correspondante.
3. Quel est à votre avis le plan d'exécution dans s'il n'existe que deux index, un sur FILM(ID-réalisateur), and un sur ARTISTE(ID-artiste)ÿ?
4. Idem, avec un index sur $FILM(ID-Film)$, et un sur $JOUE(ID-Artiste)$.
5. Idem, avec un index sur $FILM(ID-Film)$, et un sur $JOUE(ID-Film)$.

Solution :

1.
```
SELECT nom, titre
FROM   artiste, film, joue
WHERE  artiste.ID-artiste = joue.ID-artiste
AND    film.ID-film = joue.ID-film
```

2. $Artiste \bowtie (Film \bowtie Joue))$

3.
```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 NESTED LOOPS
        4 TABLE ACCESS FULL JOUE
        5 TABLE ACCESS BY ROWID ARTISTE
          6 INDEX UNIQUE SCAN ARTISTE_IDX
    7 SORT JOIN
      8 TABLE ACCESS FULL FILM
```

Comme il n'y a qu'un seul index utilisable, on fait un parcours séquentiel sur Joue pour utiliser l'index le plus tôt possible, puis on fait un tri fusion. On peut aussi faire le parcours séquentiel initial sur Film. Petit piège : l'index sur ID_réalisateur n'est pas utilisable pour la jointure.

4.
```
0 SELECT STATEMENT
  1 NESTED LOOPS
    2 NESTED LOOPS
      3 TABLE ACCESS FULL ARTISTE
      4 TABLE ACCESS BY ROWID JOUE
        5 INDEX RANGE SCAN JOUE_ARTISTE
    6 TABLE ACCESS BY ROWID FILM
      7 INDEX UNIQUE SCAN FILM_IDX
```

Comme il y a une seule table (ARTISTE) sans index propre à la jointure, on la choisit pour effectuer le parcours séquentiel initial.

5.
```
0 SELECT STATEMENT
  1 MERGE JOIN
    2 SORT JOIN
      3 NESTED LOOPS
        4 TABLE ACCESS FULL JOUE
        5 TABLE ACCESS BY ROWID FILM
          6 INDEX UNIQUE SCAN FILM_IDX
```

```
    7 SORT JOIN
      8 TABLE ACCESS FULL ARTISTE
```

Les index existant ne peuvent servir qu'à une seule jointure : celle entre JOUE et FILM. Donc il ne reste que la solution de faire un TRI-FUSION pour la jointure avec ARTISTE. NB : on parcours JOUE puis on accède à FILM par l'index. On pourrait faire le contraire (parcourir FILM et accéder à JOUE). Quand il a des statistiques, le SGBD choisit la plus petite des tables pour le parcours séquentiel.

Chapitre 12

Concurrence

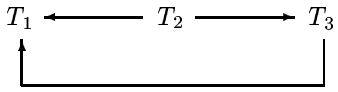
12.1 Sérialisabilité et recouvrabilité

12.1.1 Graphe de sérialisabilité et équivalence des exécutions

Construisez les graphes de sérialisabilité pour les exécutions (histoires) suivantes. Indiquez les exécutions sérialisables et vérifiez s'il y a des exécutions équivalentes.

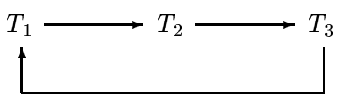
1. $H_1 : w_2[x]\ w_3[z]\ w_2[y]\ c_2\ r_1[x]\ w_1[z]\ c_1\ r_3[y]\ c_3$

   Solution :
   les conflits
   sur x: $w_2[x]\ r_1[x]$; *sur y:* $w_2[y]\ r_3[y]$; *sur z:* $w_3[z]\ w_1[z]$
   le graphe de sérialisabilité

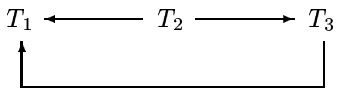
2. $H_2 : r_1[x]\ w_2[y]\ r_3[y]\ w_3[z]\ c_3\ w_1[z]\ c_1\ w_2[x]\ c_2$

   Solution :
   les conflits
   sur x: $r_1[x]\ w_2[x]$; *sur y:* $w_2[y]\ r_3[y]$; *sur z:* $w_3[z]\ w_1[z]$
   le graphe de sérialisabilité

3. $H_3 : w_3[z]\ w_1[z]\ w_2[y]\ w_2[x]\ c_2\ r_3[y]\ c_3\ r_1[x]\ c_1$

   Solution :
   les conflits
   sur x: $w_2[x]\ r_1[x]$; *sur y:* $w_2[y]\ r_3[y]$; *sur z:* $w_3[z]\ w_1[z]$
   le graphe de sérialisabilité



Conclusion: $H_1 \not\equiv H_3$, H_1 et H_3 sérialisables,

Les histoires H_1 et H_3 ne sont pas équivalentes. Pour avoir équivalence, deux conditions sont nécessaires: *(i)* avoir les mêmes transactions et les mêmes operations, et *(ii)* avoir le meme ordre des operations conflictuelles.

Ici la seconde condition est remplie, mais pas la premiere! En effet, si on extrait la transaction T_1 de ces histoires, on remarque que pour H_1 on a $T_1 = r_1[x]w_1[z]c_1$, tandis que pour H_3, $T_1 = w_1[z]r_1[x]c_1$. Et c'est pareil pour T_2.

12.1.2 Recouvrabilité

Parmi les exécutions (histoires) suivantes, lesquelles sont recouvrables, lesquelles évitent les annulations en cascade et lesquelles sont strictes? Indiquez s'il y a des exécutions sérialisables.

1. $H_1 : r_1[x]\ w_2[y]\ r_1[y]\ w_1[x]\ c_1\ r_2[x]\ w_2[x]\ c_2$

   Solution :
   T_1 lit y de T_2 ($r_1[y]$ après $w_2[y]$), mais T_1 se termine avant T_2 $\Longrightarrow$ H_1 non-recouvrable
   H_1 n'est pas sérialisable (les conflits $w_2[y]$ - $r_1[y]$ et $w_1[x]$ - $r_2[x]$ forment un cycle)

2. $H_2 : r_1[x]\ w_1[y]\ r_2[y]\ c_1\ w_2[x]\ c_2$

   Solution :
   T_2 lit y de T_1 ($r_2[y]$ après $w_1[y]$) avant que T_1 se termine $\Longrightarrow$ H_2 recouvrable, mais n'évite pas les avortements en cascade
   H_2 est sérialisable (les seuls conflits sont $r_1[x]$ - $w_2[x]$ et $w_1[y]$ - $r_2[y]$)

3. $H_3 : r_1[y]\ w_2[x]\ r_2[y]\ w_1[x]\ c_2\ r_1[x]\ c_1$

   Solution :
   T_1 lit x de T_2 ($r_1[x]$ après $w_2[x]$) après la fin de T_2, mais écrit x ($w_1[x]$ après $w_2[x]$) avant la fin de T_2 $\Longrightarrow$ H_3 évite les annulations en cascade, mais n'est pas stricte
   H_3 est sérialisable (les seuls conflits sont $w_2[x]$ - $w_1[x]$ et $w_2[x]$ - $r_1[x]$)

12.2 Contrôle de concurrence

12.2.1 Verrouillage à 2 phases

Un scheduler avec verrouillage à 2 phases reçoit la séquence d'opérations ci-dessous.

$H : r_1[x]\ r_2[y]\ w_3[x]\ w_1[y]\ w_1[x]\ w_2[y]\ c_2\ r_3[y]\ r_1[y]\ c_1\ w_3[y]\ c_3$

Indiquez l'ordre d'exécution établi par le scheduler, en considérant qu'une opération bloquée en attente d'un verrou est exécutée en priorité dès que le verrou devient disponible. On suppose que les verrous d'une transaction sont relâchés au moment du Commit.

Solution :

• $r_1[x]$, $r_2[y]$ *exécutées*

• $w_3[x]$ *bloquée à cause de* $r_1[x]$

• $w_1[y]$ *bloquée à cause de* $r_2[y]$

L'opération $w_1[y]$ *qui est bloquée va aussi bloquer tout le reste de la transaction* T_1*! Donc* $w_1[x]$ *ne peut pas s'exécuter, même si cette opération n'a pas de problème de verrou :*

• $w_1[x]$ *bloquée à cause de* $w_1[y]$

• $w_2[y]$ *exécutée*

• c_2 *relâche les verrous sur* $y \Rightarrow w_1[y]$, $w_1[x]$ *peuvent s'exécuter*

• $r_3[y]$ *bloquée car* T_3 *bloquée à cause de* $w_3[x]$

• $r_1[y]$ *exécutée*

• c_1 *relâche les verrous sur* x, $y \Rightarrow w_3[x]$, $r_3[y]$ *peuvent s'exécuter*

• $w_3[y]$, c_3 *exécutées*

Résultat*:* $r_1[x]\ r_2[y]\ w_2[y]\ c_2\ w_1[y]\ w_1[x]\ r_1[y]\ c_1\ w_3[x]\ r_3[y]\ w_3[y]\ c_3$

12.2.2 Estampillage et la règle de Thomas

Etant donée la séquence d'opérations suivante, comparez les exécutions établies par un scheduler avec estampillage simple et un scheduler intégré pur utilisant la règle de Thomas. Le scheduler avec estampillage n'utilise que le test des estampilles, sans retarder ensuite l'exécution des opérations. Considérez qu'une transaction rejetée est relancée tout de suite avec une nouvelle estampille et que ses opérations déjà exécutées sont traitées avec priorité.

H : $r_1[x]\ r_2[y]\ w_3[x]\ w_1[x]\ w_3[y]\ w_2[y]$

Solution :

Estampillage*:*

• $r_1[x]$, $r_2[y]$, $w_3[x]$ *exécutées*

• $w_1[x]$ *rejetée à cause de* $w_3[x] \Rightarrow T_1$ *rejetée et relancée en tant que* $T_4 \Rightarrow r_4[x]$, $w_4[x]$ *exécutées*

• $w_3[y]$ *exécutée*

• $w_2[y]$ *rejetée à cause de* $w_3[y] \Rightarrow T_2$ *rejetée et relancée en tant que* $T_5 \Rightarrow r_5[y]$, $w_5[y]$ *exécutées*

Résultat*:* $r_1[x]\ r_2[y]\ w_3[x]\ a_1\ r_4[x]\ w_4[x]\ w_3[y]\ a_2\ r_5[y]\ w_5[y]$

Scheduler intégré avec la règle de Thomas*:*

• $r_1[x]$, $r_2[y]$, $w_3[x]$ *exécutées*

• $w_1[x]$ *ignorée à cause de* $w_3[x]$ *(règle de Thomas)*

• $w_3[y]$ *exécutée*

• $w_2[y]$ *ignorée à cause de* $w_3[y]$ *(règle de Thomas)*

Résultat*:* $r_1[x]\ r_2[y]\ w_3[x]\ w_3[y]$

12.2.3 Comparaison des méthodes de contrôle de concurrence

Parmi les exécutions suivantes, lesquelles ne peuvent pas être obtenues par verrouillage à 2 phases et lesquelles ne peuvent pas être obtenues par estampillage simple?

H_1 : $r_1[x]\ r_2[y]\ w_2[x]\ c_2\ r_1[y]\ c_1$

Solution :

verrouillage*: impossible, car pour exécuter* $w_2[x]$,T_1 *doit relâcher le verrou obtenu par* $r_1[x]$*, donc ne pourra plus obtenir le verrou pour* $r_1[y]$

estampillage*: possible*

H_2 : $r_1[x]\ w_2[y]\ r_2[x]\ c_2\ w_1[y]\ c_1$

Solution :

verrouillage*: possible*

estampillage*: impossible, car* $w_1[y]$ *rejetée à cause de* $w_2[y]$

12.3 Reprise après panne

12.3.1 Journalisation

Soit le journal physique ci-dessous, dans lequel on a marqué les opérations Commit et Abort réalisées:

$[T_1,x,3],[T_2,y,1],[T_1,z,2],c_1,[T_2,z,4],[T_3,x,2],a_2,[T_4,y,3],c_3,[T_5,x,5]$

1. Indiquez le contenu de *liste_commit*, *liste_abort*, *liste_active*.

   Solution :
   liste_commit = $\{T_1,T_3\}$*;* liste_abort = $\{T_2\}$*;* liste_active = $\{T_4,T_5\}$*;*

2. En supposant qu'une nouvelle écriture vient de s'ajouter au journal, lesquelles des écritures suivantes sont compatibles avec une exécution stricte: $[T_5,y,4]$, $[T_4,z,3]$ ou $[T_6,x,1]$?

   Solution :
   $[T_5,y,4]$ *écrit* y*; y déjà écrit par* T_4 *(encore active)* $\Rightarrow$ *exécution non-stricte*
   $[T_4,z,3]$ *écrit* z*; z déjà écrit par* T_1 *et* T_2*, déjà terminées* $\Rightarrow$ *exécution stricte*
   $[T_6,x,1]$ *écrit* x*; x déjà écrit par* T_5 *(encore active)* $\Rightarrow$ *exécution non-stricte*

3. Quelles sont les entrées récupérables par l'algorithme de ramasse-miettes?

   Solution :
   $[T_2,y,1]$, $[T_2,z,4]$ *récupérables, car* T_2 *rejetée*
   $[T_1,x,3]$ *récupérable, car* T_3 *validée a écrit ensuite* x

4. Si les valeurs initiales des enregistrements étaient $x = 1$, $y = 2$,$z = 3$, et si une panne survenait à ce moment, quelles seraient les valeurs restaurées pour x, y et z après la reprise?

   Solution :
   T_3 est la dernière transaction validée à avoir écrit dans x ($[T_3,x,2]$), donc la valeur restaurée est x=2.
   Aucune transaction validée n'a écrit dans y, donc la valeur restaurée est y=2, valeur initiale.
   T_1 est la dernière transaction validée à avoir écrit dans z ($[T_1,z,2]$), donc la valeur restaurée est z=2.

12.4 Concurrence: Gestion Bancaire

Les trois programmes suivants peuvent s'exécuter dans un système de gestion bancaire. *Débit* diminue le solde d'un compte `c` avec un montant donné `m`. Pour simplifier, tout débit est permis (on accepte des découverts). *Crédit* augmente le solde d'un compte `c` avec un montant donné `m`. *Transfert* transfère un montant `m` à partir d'un compte source `s` vers un compte destination `d`. L'exécution de chaque programme démarre par un **Start** et se termine par un **Commit** (non montrés ci-dessous).

```
Débit (c:Compte;      |    Crédit (c:Compte;      |    Transfert (s,d:Compte;
       m:Montant)     |            m:Montant)     |               m:Montant)
begin                 |    begin                  |    begin
  t := Read(c);       |      t = Read(c);         |      Débit(s,m);
  Write(c,t-m);       |      Write(c,t+m);        |      Crédit(d,m);
end                   |    end                    |    end
```

Le système exécute en même temps les trois opérations suivantes:

(1) un transfert de montant 100 du compte A vers le compte B
(2) un crédit de 200 pour le compte A
(3) un débit de 50 pour le compte B

1. Écrire les transactions T_1, T_2 et T_3 qui correspondent à ces opérations. Montrer que l'histoire **H**: $r_1[A]$ $r_3[B]$ $w_1[A]$ $r_2[A]$ $w_3[B]$ $r_1[B]$ c_3 $w_2[A]$ c_2 $w_1[B]$ c_1 est une exécution concurrente de T_1, T_2 et T_3.

   Solution :
   `Débit` *et* `Crédit` *sont constitués chacun d'une lecture, suivie d'une écriture. Dans ce cas, les transactions T_1, T_2 et T_3 seront:*

   T_1: $r_1[A]$ $w_1[A]$ $r_1[B]$ $w_1[B]$ c_1
   T_2: $r_2[A]$ $w_2[A]$ c_2
   T_3: $r_3[B]$ $w_3[B]$ c_3

   L'histoire **H** *contient toutes les opérations de T_1, T_2 et T_3 et respecte l'ordre des opérations dans chaque transaction. Donc* **H** *est une exécution concurrente de T_1, T_2 et T_3.*

2. Mettre en évidence les conflits dans **H** et construire le graphe de sérialisation de cette histoire. **H** est-elle sérialisable? **H** est-elle recouvrable?

   Solution :
   Les conflits sur A: $r_1[A]$-$w_2[A]$;$w_1[A]$-$r_2[A]$; $w_1[A]$-$w_2[A]$
   Les conflits sur B: $r_3[B]$-$w_1[B]$; $w_3[B]$-$r_1[B]$; $w_3[B]$-$w_1[B]$
   Le graphe de sérialisation **SG(H)**: $T_3 \rightarrow T_1 \rightarrow T_2$
   H *est sérialisable, car le graphe ne contient pas de cycle.*
   H *n'est pas recouvrable, car T_2 lit A de T_1 (après $w_1[A]$ on a $r_2[A]$), mais T_2 se termine avant T_1. La même conclusion est obtenue en considérant la suite $w_3[B]$ $r_1[B]$.*

3. Quelle est l'exécution **H'** obtenue à partir de **H** par verrouillage à deux phases? On suppose que les verrous d'une transaction sont relâchés après le Commit de celle-ci. Une opération bloquée en attente d'un verrou bloque le reste de sa transaction. Au moment du relâchement des verrous, les opérations en attente sont exécutées en priorité.
   Si au début le compte A avait un solde de 100 et B de 50, quel sera le solde des deux comptes après la reprise si une panne intervient après l'exécution de $w_1[B]$?

   Solution :
   $r_1[A]$, $r_3[B]$ *reçoivent les verrous de lecture et s'exécutent*
   $w_1[A]$ *obtient le verrou d'écriture sur A (déjà obtenu en lecture par* T_1*) et s'exécute*
   $r_2[A]$ *bloquée en attente de verrou sur A* $\Rightarrow$ T_2 *bloquée*
   $w_3[B]$ *obtient le verrou d'écriture sur B (déjà obtenu en lecture par* T_3*) et s'exécute*
   $r_1[B]$ *bloquée en attente de verrou sur B* $\Rightarrow$ T_1 *bloquée*
   c_3 *s'exécute et relâche les verrous sur B* $\Rightarrow$ $r_1[B]$ *débloquée, obtient le verrou et s'exécute (*T_1 *débloquée)*
   $w_2[A]$ *et* c_2 *bloquées car* T_2 *bloquée*
   $w_1[B]$ *obtient le verrou et s'exécute*
   c_1 *s'exécute et relâche les verrous sur A* $\Rightarrow$ $r_2[A]$, $w_2[A]$ *et* c_2 *s'exécutent.*
   Le résultat est **H'***:* $r_1[A]$ $r_3[B]$ $w_1[A]$ $w_3[B]$ c_3 $r_1[B]$ $w_1[B]$ c_1 $r_2[A]$ $w_2[A]$ c_2
   Si une panne intervient après l'exécution de $w_1[B]$*, seule la transaction* T_3 *(le débit de 50 sur B) est validée à ce moment. Après la reprise, seul l'effet de* T_3 *sera retrouvé, donc le compte A aura un solde de 100 et B de 0.*