Ce document constitue l'annexe A de l'ouvrage "Bases de données", J-L Hainaut, Dunod, 2009

***Date de dernière modification** : 7/5/2010*

Annexe **A**

Exercices et solutions

Cette annexe propose une collection d'exercices, certains assortis d'une suggestion de solution, et classés selon les chapitres de l'ouvrage. Elle reprend intégralement les exercices qui apparaissent en fin de chapitres.
Les solutions sont données à titre indicatif et de bonne foi. L'auteur ne peut en aucune manière garantir qu'elles sont ni correctes ni, quand bien même elles le seraient, qu'elles sont les meilleures ou qu'elles sont appropriées aux besoins spécifiques du lecteur.

A.1 CHAPITRE 1 - MOTIVATION ET INTRODUCTION

Néant

A.2 CHAPITRE 2 - CONCEPTS DES BASES DE DONNÉES

2.1 On considère le bon de commande papier de la figure 2.1, qu'on se propose d'encoder sous la forme de données à introduire dans la base de données de la figure 2.8. Qu'en pensez-vous ?

Solution

Les données de ce bon de commande présentent plusieurs anomalies qui en empêcheront l'introduction dans la base de données.

Numéro de commande déjà présent dans la BD. Violation d'une contrainte d'unicité.

Date de commande invalide. Violation du domaine de valeurs.

Numéro de client inexistant. Violation d'une contrainte référentielle.

Adresse du client manquante. Violation du caractère obligatoire d'une colonne.

Commande N°: ***30186*** Date : ***30/2/2009***

Numéro client ***B516***
Nom ***ASSRAN***
Adresse
Localité ***Cassis***

N°PRODUIT	LIBELLE PRODUIT	PRIX	QUANTITE	SOUS-TOTAL
PA45	***POINTE ACIER 45 (20K)***	***105***	***un***	***105***
PA45	***POINTE ACIER 45 (20K)***	***95***	***trois***	***285***
			TOTAL COMMANDE	***422***

Figure 2.1 - Un bon de commande curieux

Deux détails référencent le même produit. Violation d'une contrainte d'unicité (identifiant de DETAIL).

Les quantités sont exprimées en caractères. Violation du domaine de valeurs.

Le *produit PA45 possède deux prix*. Violation d'une dépendance fonctionnelle.

Le montant total est incorrect. Sans importance, il s'agit d'une donnée calculée non enregistrée.

2.2 Vérifier si le schéma ci-dessous est normalisé. Si nécessaire, le décomposer en tables normalisées.

CLIENT ⟶ ADRESSE, DELEGUE
DELEGUE ⟶ REGION

Solution

La colonne REGION dépend d'une colonne qui n'est pas un identifiant. La table n'est pas normalisée. On la décompose en deux tables VENTE(NPRO, CLIENT, DATE, QUANTITE, ADRESSE, DELEGUE) et REP(DELEGUE, REGION). Ensuite, dans la nouvelle table VENTE, les colonnes ADRESSE et DELEGUE dépendent d'une colonne qui n'est pas un identifiant. Par décomposition, on obtient le schéma ci-dessous :

VENTE(NPRO, CLIENT, DATE, QUANTITE)

CLI(CLIENT, ADRESSE, DELEGUE)

REP(DELEGUE, REGION)

Deux clés étrangères : CLIENT de VENTE et DELEGUE de CLI.

2.3 Décomposer si nécessaire la table ci-dessous.

NCLI ⟶ NOM
NPRO ⟶ LIBELLE

Solution

La colonne NOM dépend d'une colonne qui n'est pas un identifiant. La table n'est pas normalisée. On la décompose en deux tables COMMANDE(NCOM, NCLI, DATE, NPRO, LIBELLE) et CLIENT(NCLI, NOM). Ensuite, dans la nouvelle table COMMANDE, la colonnes LIBELLE dépend d'une colonne qui n'est pas un identifiant. Par décomposition, on obtient le schéma ci-dessous :

COMMANDE(NCOM, NCLI, DATE, NPRO)

CLIENT(NCLI, NOM)

PRODUIT(NPRO, LIBELLE)

Deux clés étrangères : NCLI de COMMANDE et NPRO de COMMANDE.

2.4 Décomposer si nécessaire la table ci-dessous.

DATE_INTRO, IMPORTATEUR ⟶ AGREATION

Solution

La colonne AGREATION dépend de colonnes qui ne forment pas un identifiant. La table n'est pas normalisée. On la décompose en deux tables PRODUIT(NPRO, DATE_INTRO, IMPORTATEUR) et AGRE(DATE_INTRO, IMPORTATEUR, AGREATION). Une clé étrangère : (DATE_INTRO, IMPORTATEUR) de PRODUIT.

A.3 CHAPITRE 3 - MODÈLE RELATIONNEL ET NORMALISATION

3.1 Décomposer si nécessaire la relation ACHAT.

ACHAT(NCOM, NPRO, PRIX)
NCOM ⟶ NPRO
NPRO ⟶ PRIX

Solution

L'identifiant de ACHAT est {NCOM}. La DF NPRO ⟶ PRIX est donc anormale. Par décomposition selon cette DF, on obtient le schéma relationnel normalisé :

ACHAT(NCOM, NPRO); PRODUIT(NPRO, PRIX);
ACHAT[NPRO] ⊆ PRODUIT[NPRO]

3.2 Décomposer si nécessaire la relation COMMANDE.

COMMANDE(NCOM, NCLI, NOM, DATE, NPRO, LIBELLE)
NCOM ⟶ NCLI, DATE, NPRO
NCLI ⟶ NOM
NPRO ⟶ LIBELLE

Solution

L'identifiant de COMMANDE est {NCOM}. Les DF NCLI ⟶ NOM et NPRO ⟶ LIBELLE sont donc anormales. Par décomposition selon chacune de ces DF, on obtient le schéma relationnel normalisé :

COMMANDE(NCOM, NCLI, DATE, NPRO);
CLIENT(NCLI, NOM); PRODUIT(NPRO, LIBELLE);
COMMANDE[NCLI] ⊆ CLIENT[NCLI]
COMMANDE[NPRO] ⊆ PRODUIT[NPRO]

3.3 Décomposer si nécessaire la relation ACHAT2.

ACHAT2(CLI, PRO, MAG, PRIX)
PRO, MAG ⟶ PRIX

Solution

L'identifiant de ACHAT2 est {CLI, PRO, MAG}. La DF PRO, MAG ⟶ PRIX est donc anormale. On obtient par décomposition :

ACHAT2(CLI, PRO, MAG); TARIF(PRO, MAG, PRIX));
ACHAT2[PRO, MAG] ⊆ TARIF[PRO, MAG]

3.4 Décomposer si nécessaire la relation ACHAT3.

ACHAT3(CLI, PRO, MAG, PRIX)
CLI, PRO, MAG ⟶ PRIX

Solution

L'identifiant de la relation ACHAT3 est {CLI, PRO, MAG}. Celle-ci est donc normalisée.

3.5 Décomposer si nécessaire la relation ECRIT (POSITION indique la position de l'auteur dans la liste des auteurs).

ECRIT(AUTEUR, OUVRAGE, POSITION)
AUTEUR, OUVRAGE ⟶ POSITION
OUVRAGE, POSITION ⟶ AUTEUR

Solution

Le graphe ADF comporte un circuit. Les identifiants de la relation ECRIT sont {AUTEUR, OUVRAGE} et {OUVRAGE, RANG}. Celle-ci est normalisée.

3.6 Calculer les identifiants de la relation CINE. Décomposer cette relation si nécessaire.

CINE(FILM, VILLE, SALLE, DISTRIBUTEUR, DELEGUE)
SALLE ⟶ VILLE
FILM, VILLE ⟶ SALLE, DISTRIBUTEUR
DISTRIBUTEUR ⟶ DELEGUE

Solution

Le graphe ADF comporte un circuit. Les identifiants sont {FILM, VILLE} et {SALLE, FILM}. Les deux DF suivantes sont donc anormales : SALLE ⟶ VILLE et DISTRIBUTEUR ⟶ DELEGUE. Cette dernière étant externe, elle permet une première décomposition :

CINE(<u>FILM, VILLE</u>, SALLE, DISTRIBUTEUR);
DIS(<u>DISTRIBUTEUR</u>, DELEGUE);
CINE[DISTRIBUTEUR] ⊆ DIS[DISTRIBUTEUR]
SALLE ⟶ VILLE
FILM, VILLE ⟶ DISTRIBUTEUR

La DF FILM, VILLE ⟶ DISTRIBUTEUR, non anormale, est externe et ne fait pas partie du noyau irréductible. Elle peut donc faire l'objet d'une décomposition :

CINE(<u>FILM, VILLE</u>, SALLE);
DISTR(<u>FILM, VILLE</u>, DISTRIBUTEUR);
DIS_DEL(<u>DISTRIBUTEUR</u>, DELEGUE);
CINE[FILM, VILLE] ⊆ DISTR[FILM, VILLE]
DISTR[DISTRIBUTEUR] ⊆ DIS_DEL[DISTRIBUTEUR]
SALLE ⟶ VILLE

Le noyau résiduel {FILM, VILLE, SALLE} est irréductible et non normalisé. Selon le canevas 3.8.5, la dernière relation CINE peut être remplacée par un des trois schémas ci-dessous :

1. CINE(<u>FILM, VILLE</u>, SALLE); SALLE ⟶ VILLE
2. CINE(FILM, SALLE); LOC(<u>SALLE</u>, VILLE);

CINE[SALLE] = LOC[SALLE]
CINE*LOC: FILM, VILLE ⟶ SALLE

3. CINE(FILM, VILLE, SALLE); LOC(SALLE, VILLE);
CINE[SALLE, VILLE] = LOC[SALLE, VILLE]

3.7 La version populaire des règles d'Armstrong en comporte une sixième, la *pseudo-transitivité*, qui s'énonce comme suit.

Si on a K ⟶ L et LA ⟶ M, on a aussi KA ⟶ M.

Démontrez que cette règle est dérivable des autres.

Solution

Par réflexivité, on a A ⟶ A. Par additivité, K ⟶ L et A ⟶ A donnent KA ⟶ LA. Par transitivité, KA ⟶ LA et LA ⟶ M donnent KA ⟶ M. CQFD

3.8 Décomposer si nécessaire la relation VENTE.

VENTE(NPRO, CLIENT, DATE, QUANTITE, ADRESSE, DELEGUE, REGION)
NPRO, CLIENT, DATE ⟶ QUANTITE
CLIENT ⟶ ADRESSE, DELEGUE
DELEGUE ⟶ REGION

3.9 Décomposer si nécessaire la relation PRODUIT.

PRODUIT(NPRO, DATE-INTRO, IMPORTATEUR, AGREATION)
NPRO ⟶ DATE-INTRO, IMPORTATEUR
DATE-INTRO, IMPORTATEUR ⟶ AGREATION

3.10 Décomposer si nécessaire la relation VOYAGE.

VOYAGE(NUMV, NUMC, DATE, MODELE, NOM)
NUMC ⟶ NOM
NUMV ⟶ MODELE

3.11 Calculer les identifiants de la relation PROJET. Décomposer cette relation si nécessaire.

PROJET(CODE, TITRE, NUM-CONTRAT, BUDGET, RESPONSABLE, UNITE)
CODE ⟶ TITRE, BUDGET
NUM-CONTRAT ⟶ CODE, RESPONSABLE
TITRE ⟶ NUM-CONTRAT, UNITE

Solution

Le graphe ADF comporte un circuit comprenant les attributs {CODE, NUM-CONTRAT, TITRE}. Les identifiants sont {CODE}, {NUM-CONTRAT} et {TITRE}. Chacun des déterminants est un identifiant. La relation PROJET est donc normalisée.

3.12 Calculer les identifiants de la relation ACHAT4. Décomposer cette relation si nécessaire.

ACHAT4(CLIENT, FOURN, ADR-F, ARTICLE, PRIX, DELAI)
CLIENT, ARTICLE ⟶ FOURN, PRIX
FOURN ⟶ ARTICLE, ADR-F
ARTICLE, FOURN ⟶ DELAI

Solution

Identifiants : {CLIENT, ARTICLE} et {CLIENT, FOURN}. Il existe des DF anormales rendant la relation ACHAT4 non normalisée.

Dépendances de base : on observe que la DF ARTICLE, FOURN ⟶ DELAI n'est pas minimale; il faut la réduire à FOURN ⟶ DELAI, ce qui va simplifier les choses. On réécrit donc l'énoncé comme suit :

ACHAT4(CLIENT, FOURN, ADR-F, ARTICLE, PRIX, DELAI)
CLIENT, ARTICLE ⟶ FOURN, PRIX
FOURN ⟶ ADR-F, ARTICLE, DELAI

On conserve des contraintes d'égalité lors des décompositions. On rectifiera à la fin si nécessaire.

0) Première passe

R1(CLIENT, ARTICLE, PRIX)
R2(FOURN, ADR-F)
R3(FOURN, DELAI)
R4(CLIENT, ARTICLE, FOURN)
R4: FOURN⟶ ARTICLE
R2[FOURN] = R3[FOURN] = R4[FOURN]
R4[CLIENT, ARTICLE] = R1[CLIENT, ARTICLE]

R4 constitue un noyau irréductible non normalisé.

1) La peste (3FN)

R23(FOURN, ADR-F, DELAI)
R14(CLIENT, ARTICLE, PRIX, FOURN)
R14: FOURN ⟶ ARTICLE
R14[FOURN] = R23[FOURN]

2) Le choléra (FNBC)

R1(CLIENT, ARTICLE, PRIX)
R2(FOURN, ADR-F)
R3(FOURN, DELAI)
R4'(FOURN, ARTICLE)
R4''(CLIENT, FOURN)
R4'*R4'': CLIENT, ARTICLE ⟶ FOURN
R2[FOURN] = R3[FOURN] = R4'[FOURN] = R4''[FOURN]
R4'*R4''[CLIENT, ARTICLE] = R1[CLIENT, ARTICLE]
Cette dernière contrainte dérive directement de celle du cas (1)

Les contraintes d'égalité nous autorisent à simplifier ce schéma comme suit :

R1(CLIENT, ARTICLE, PRIX)
R234'(FOURN, ADR-F, DELAI, ARTICLE)
R4''(CLIENT, FOURN)
R4'*R234'': CLIENT, ARTICLE ⟶ FOURN
R234'[FOURN] = R4''[FOURN]
R234'*R4''[CLIENT, ARTICLE] = R1[CLIENT, ARTICLE]

3) La peste et le choléra (FNCE)

R23(FOURN, ADR-F, DELAI)
R14(CLIENT, ARTICLE, PRIX, FOURN)
R4'(FOURN, ARTICLE)
R14[FOURN] = R23[FOURN]
R14[FOURN, ARTICLE] = R4'[FOURN, ARTICLE]

Les contraintes d'égalité nous autorisent à simplifier ce schéma comme suit :

R234'(FOURN, ADR-F, DELAI, ARTICLE)
R14(CLIENT, ARTICLE, PRIX, FOURN)
R14[FOURN, ARTICLE] = R234'[FOURN, ARTICLE]

4) Clôture

Il reste à attribuer des noms significatifs aux relations et à préciser les contraintes d'inclusion.

3.13 En vous servant des propriétés des contraintes d'inclusion, affinez les définitions suivantes :

OFFRE(PRODUIT, FOURN)
COMMANDE(CLIENT, PRODUIT, FOURN, DATE, QTE)
COMMANDE[PRODUIT, FOURN] ⊆ LIVRE[PRODUIT, FOURN]

3.14 On considère une base de données comportant les deux relations

PAYS(NOM, CAPITALE)
VILLE(NOM, PAYS)

PAYS reprend pour chaque pays son nom et celui de sa capitale tandis que VILLE reprend pour chaque ville son nom et celui de son pays. Sachant qu'il n'y a pas deux pays de même nom, ni deux villes de même nom dans un même pays, complétez le schéma de cette base de données.

3.15 Selon les propriétés des contraintes d'inclusion, la relation CLIENT(NCLI, NOM, ADRESSE, LOCALITE) ne contient-elle pas une clé étrangère ?

3.16 Lorsque vous aurez maîtrisé le langage SQL DDL (chapitre 5), si par hasard vous repassez par ici, essayez de traduire en SQL les structures de la solution *c* (*La peste **et** le choléra*) de la section 3.8.5.

3.17 En analysant les données de la relation ci-dessous, déterminer les dépendances fonctionnelles dont elle est le siège. Parmi celles-ci, quelles sont celles qui semblent pertinentes ?

NumPiece	Description	Fournisseur	AdresseFournisseur	Prix
10010	750 GB Disk	Seagate	Washington	120
10010	750 GB Disk	Samsung	Versailles	120
10220	2 GB RAM card	Kensington	Londres	95
10220	2 GB RAM card	Samsung	Versailles	100
10220	2 GB RAM card	Sun Microsystems	Palo Alto	100
10440	21" LCD Monitor	Samsung	Versailles	310

3.18 Un jeune lecteur de Carpentras nous écrit pour nous faire part de ses doutes sur le procédé de normalisation décrit à la section 3.8.4. Il estime par exemple que le schéma suivant :

FAB(USINE, PRODUIT, ADRESSE, DESCRIPTION)
USINE ⟶ ADRESSE
PRODUIT ⟶ DESCRIPTION

peut tout aussi bien se décomposer comme suit[1] :

U(USINE, ADRESSE)
P(PRODUIT, DESCRIPTION)
U*P: USINE, PRODUIT ⟶ ADRESSE, DESCRIPTION

Qu'en pensent les autres lecteurs ?

1. La jointure U*P, pour laquelle on ne précise pas les colonnes de jointure, est un *produit relationnel*. Cet opérateur, qui sera décrit à la section 7.4, correspond à une jointure naturelle dans laquelle il n'existerait pas de condition de jointure. Chaque n-uplet de U est associé à chaque n-uplet de P.

A.4 CHAPITRE 4 - IMPLÉMENTATION DES STRUCTURES DE DONNÉES

4.1 On considère une table dont les lignes sont rangées dans un espace qui lui est exclusivement réservé et constitué de pages de 4 Ko. Chaque ligne occupe une seule page. Cet espace est implanté sur un disque de notre modèle de référence (figure 4.5). La table contient 1.000.000 lignes d'une longueur de 200 octets. Le taux d'occupation moyen des pages est de 75%. Calculer le volume minimal de cet espace de stockage et le temps de lecture séquentielle de toutes les lignes de cette table.

Solution

Capacité d'une page : $\mathbf{Mrpp} = \lfloor \mathbf{Lp} / \mathbf{Lr} \rfloor = 20$ enregistrements par page.

Contenu effectif d'une page : $\mathbf{Nrpp} = \tau \times \mathbf{Mrpp} = 15$ enregistrements par page.

Volume minimal de l'espace : $\mathbf{Np} = \lceil \mathbf{Nr} / \mathbf{Nrpp} \rceil =$ **66.667** pages ou **261** Mo.

Temps de lecture séquentielle de la table (hypothèse de lecture anticipée d'une piste, soit $\mathbf{t_{ls1}} = 0{,}145$ msec par page) : $\mathbf{t_{ls}} = \mathbf{t_{ls1}} \times \mathbf{Np} =$ **9,67** sec.

4.2 Une table APPEL, représentant des appels téléphoniques, comporte 6.000.000 lignes d'une longueur fixe de 200 octets. Elle dispose d'un identifiant constituée de la colonne ID de 40 caractères, sur laquelle on définit un index. La table est stockée dans un espace qui lui est réservé, implanté sur un disque du modèle de référence, décomposé en pages de 8 Ko. On envisage trois techniques d'implémentation de l'index sur ID : index primaire en séquentiel indexé, index primaire en calculé, index secondaire sur fichier en vrac. Calculer dans chaque cas le volume minimal de l'espace de stockage et le temps d'accès via ID.

Solution

On admet que dans les trois organisations le taux d'occupation moyen du fichier de base est $\tau_b = 0{,}8$. La capacité d'une page est $\mathbf{Mrpp} = \lfloor \mathbf{Lp} / \mathbf{Lr} \rfloor = 40$ enregistrements. Son contenu effectif moyen est $\mathbf{Nrpp} = \tau_b \times \mathbf{Mrpp} = 32$ enregistrements. La taille du fichier de base est donc $\mathbf{Npb} = \lceil \mathbf{Nr} / \mathbf{Nrpp} \rceil =$ **187.500** pages.

Index primaire en séquentiel indexé (SI).

Compte tenu d'un taux de remplissage des pages d'index de $\tau_i = 0{,}8$ et de pointeurs de page de 4 octets, on calcule $\mathbf{Li} = 44$ octets, $\mathbf{Mipp} = \lfloor \mathbf{Lp} / \mathbf{Li} \rfloor = 186$ entrées par page et $\mathbf{Nipp} = \tau_i \times \mathbf{Mipp} = 148{,}8$ entrées par page. On en déduit $\mathbf{Npi} = \mathbf{Npi(3)} + \mathbf{Npi(2)} + \mathbf{Npi(1)} = 1.261 + 9 + 1 = 1.271$ pages et $\mathbf{n} = 3$ (aussi calculable par $\mathbf{n} = \lceil \log_{\mathbf{Nipp}} \mathbf{Npb} \rceil = \lceil \ln \mathbf{Npb} / \ln \mathbf{Nipp} \rceil = \lceil 2{,}427 \rceil = 3$). Taille du fichier $\mathbf{Np} = \mathbf{Npb} + \mathbf{Npi}. = 187.500 + 1.271 =$ **188.771** pages. Le calcul simplifié $\mathbf{Npi} \cong \mathbf{Npi(n)}$ conduit à une erreur totalement négligeable de 10 pages, soit 0,0053%.

Le temps d'accès via l'identifiant est celui de 4 lectures de pages, soit 49,2 msec. Compte tenu d'un verrouillage des deux premiers niveaux d'index (10 pages) dans le tampon, ce temps tombe à 2 lectures de pages, soit 24,6 msec. Si le tampon peut en outre accueillir 1.261 pages supplémentaires, le temps d'accès n'est plus que de **12,3** msec.

Index primaire en calculé.

L'espace occupé par le fichier calculé se limite à celui du fichier de base, soit **187.500** pages. L'abaque nous donne, pour la courbe **Mrpp** = 40 et un taux d'occupation de 80% un coût d'accès de 1,01 à 1,02 accès physique, soit 1,015 x 12,3 = **12,5** msec. L'accès ne nécessite pas plus d'une page dans le tampon.

Index secondaire sur fichier en vrac.

Le dictionnaire de valeurs de l'index contient **Nv** = 6.000.000 entrées de **Lv** = 46 octets (un pointeur d'enregistrement de 6 octets par entrée). On a donc **Mvpp** = $\lfloor$**Lp** / **Lv**$\rfloor$ = 178 entrées par page et **Nvpp** = τ**v** x **Mvpp** = 142,4 entrées par page (τ**v** = 0,8). On en déduit la taille du dictionnaire : **Npv** = $\lceil$**Nv** / **Nvpp**$\rceil$ = 42.135 pages. On traite ensuite ce dictionnaire comme un fichier de base organisé en séquentiel indexé. Son index primaire se calcule de manière classique. On reprend les grandeurs dont nous disposons déjà : **Li** = 44; **Mipp** = 186; **Nipp** = 148,8. On a aussi **Npi** = **Npi(3)** + **Npi(2)** + **Npi(1)** = 284 + 2 + 1 = 287 pages et **n** = 3. Le volume de l'index secondaire est **Nps** = **Npv** + **Npi** = 42.442 pages. La taille totale du fichier est de **Np** = **Npb** + **Nps**. = 187.500 + 42.442 = **229.942** pages.

Le temps d'accès via l'identifiant est de 5 lectures de pages (3 pour l'index du dictionnaire + 1 pour le dictionnaire + 1 pour le fichier de base), soit 61,5 msec. Si on verrouille des deux premiers niveaux d'index du dictionnaire (3 pages) dans le tampon, ce temps est de 3 lectures de pages, soit 36,9 msec. Si le tampon contient la totalité de l'index du dictionnaire (287 pages), le temps d'accès est de 2 lectures de pages, soit de **24,6** msec. Il est difficile de diminuer ce temps.

En résumé

	volume index	**volume tampon**	**temps d'accès**
Primaire SI	10 Mo	0,08 ou 10 Mo	24,6 ou 12,3 msec
Primaire calculé	0	0,008 Mo	12,5 msec
Secondaire	332 Mo	0,024 ou 2,25 Mo	36,9 ou 24,6 msec

4.3 Soit un fichier calculé dont **Lr** = 400 octets, **Lp** = 4Ko et rempli à 90%. Combien de lectures par clé peut-on réaliser par seconde ?

Solution

La capacité des pages est de **Mrpp** = $\lfloor$**Lp** / **Lr**$\rfloor$ = 5 enregistrements. L'abaque nous donne, à l'abscisse 90, le nombre d'accès n_k = 1,29, soit un temps d'accès de n_k x t_{la1} = 15,9 msec. La vitesse de lecture est donc de 1/0,0159 = **62** enregistrements par seconde.

4.4 Quel est le taux d'occupation à partir duquel il convient de réorganiser un fichier calculé dont **Mrpp** = 20 si le temps moyen d'accès par clé ne peut dépasser 15 msec ?

Solution

Un temps de 15 msec correspond à 0,015/**tla1** = 1,22 accès physique aléatoires. Pour **Mrpp** = 20 et **nk** = 1,22 l'abaque nous donne un taux d'occupation de **96**%.

4.5 Quelle taille de page minimum préconiser pour un fichier calculé dont le temps d'accès ne peut dépasser 1,15 accès physiques au taux de remplissage de 90% ?

Solution

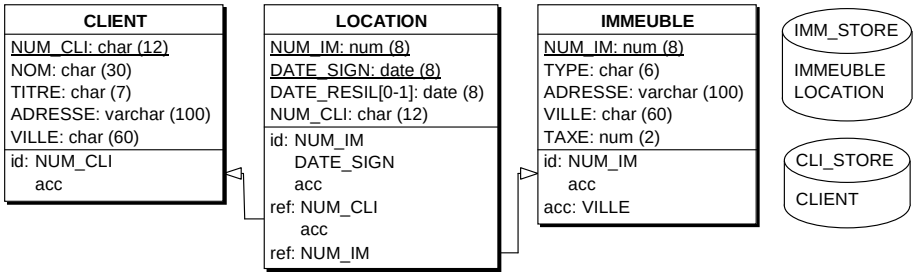
Pour ces données, l'abaque nous indique une valeur **Mrpp** comprise entre 10 et 20 mais plus proche de 20. La fonction **nk** = f(**Mrpp**) présentant une allure logarithmique inverse, on estime la taille recherchée à **15** enregistrements par page.

A.5 CHAPITRE 5 - LES SYSTÈMES DE GESTION DE BASES DE DONNÉES

Néant

A.6 CHAPITRE 6 - LE LANGAGE SQL DDL

6.1 Écrire le code SQL DDL correspondant au schéma ci-dessous.



Solution
Le code ci-dessous a été généré par l'atelier DB-MAIN selon le style *Standard SQL*. A noter que ce schéma n'est pas strictement conforme au standard SQL2.

```
-- ********************************************
-- * Standard SQL generation                  *
-- *------------------------------------------*
-- * Generator date:  Apr 14 2003             *
-- * Generation date: Sun Dec 28 18:17:20 2008 *
-- ********************************************

-- Database Section
-- ________________

create database DUNOD_2009_Ch06;

-- DBSpace Section
-- _______________

create dbspace IMM_STORE;

create dbspace CLI_STORE;

-- Table Section
-- _____________

create table CLIENT (
     NUM_CLI char(12) not null,
     NOM char(30) not null,
     TITRE char(7) not null,
     ADRESSE varchar(100) not null,
     VILLE char(60) not null,
     primary key (NUM_CLI))
     in CLI_STORE;
```

```
create table IMMEUBLE (
     NUM_IM numeric(8) not null,
     TYPE char(6) not null,
     ADRESSE varchar(100) not null,
     VILLE char(60) not null,
     TAXE numeric(2) not null,
     primary key (NUM_IM))
     in IMM_STORE;

create table LOCATION (
     NUM_IM numeric(8) not null,
     DATE_SIGN date not null,
     DATE_RESIL date,
     NUM_CLI char(12) not null,
     primary key (NUM_IM, DATE_SIGN))
     in IMM_STORE;

-- Constraints Section
-- ____________________

alter table LOCATION add constraint GRLOCATION
     foreign key (NUM_CLI)
     references CLIENT;
alter table LOCATION add constraint GRLOCATION_1
     foreign key (NUM_IM)
     references IMMEUBLE;

-- Index Section
-- _____________
create unique index IDCLIENT
     on CLIENT (NUM_CLI);
create unique index IDIMMEUBLE
     on IMMEUBLE (NUM_IM);
create index GRIMMEUBLE
     on IMMEUBLE (VILLE);
create index GRLOCATION
     on LOCATION (NUM_CLI);
create unique index IDLOCATION
     on LOCATION (NUM_IM, DATE_SIGN);
```

A.7 CHAPITRE 7 - LE LANGAGE SQL DML (1)

a) Énoncés de type 1

7.1 Afficher les caractéristiques des produits (c'est-à-dire, pour chaque produit, afficher ses caractéristiques).

```
select *
from   PRODUIT
```

7.2 Afficher la liste des localités dans lesquelles il existe au moins un client.

```
select distinct LOCALITE
from   CLIENT
```

7.3 Afficher le numéro, le nom et la localité des clients de catégorie C1 n'habitant pas à Toulouse.

```
select NCLI, NOM, LOCALITE
from   CLIENT
where  CAT = 'C1'
and    LOCALITE <> 'Toulouse'
```

7.4 Afficher les caractéristiques des produits en acier.

```
select *
from   PRODUIT
where  LIBELLE like '%ACIER%'
```

7.5 Donner le numéro, le nom et le compte des clients de Poitiers et de Bruxelles dont le compte est positif.

```
select NCLI, NOM, COMPTE
from   CLIENT
where  LOCALITE in ('Poitiers','Bruxelles')
and    COMPTE > 0
```

b) Énoncés de type 2

7.6 Quelles catégories de clients trouve-t-on à Toulouse ?

```
select distinct CAT
from   CLIENT
where  LOCALITE = 'Toulouse'
and    CAT is not null
```

7.7 Afficher le numéro, le nom et la localité des clients dont le nom précède alphabétiquement la localité où ils résident.

```
select NCLI, NOM, LOCALITE
from   CLIENT
where  NOM < LOCALITE
```

7.8 Afficher le total, le minimum, la moyenne et le maximum des comptes des clients (compte non tenu des commandes actuelles).

```
select sum(COMPTE) as somme,
       avg(COMPTE) as moyenne,
       min(COMPTE) as minimum,
       max(COMPTE) as maximum
from   CLIENT
```

7.9 Afficher les numéros des clients qui commandent le produit de numéro `'CS464'`.

```
select distinct NCLI
from   COMMANDE
where  NCOM in (select NCOM
                from   DETAIL
                where  NPRO = 'CS464')
```

7.10 Afficher les localités des clients qui commandent le produit de numéro `'CS464'`.

```
select distinct LOCALITE
from   CLIENT
where  NCLI in (select NCLI
                from   COMMANDE
                where  NCOM in (select NCOM
                                from   DETAIL
                                where  NPRO = 'CS464'))
```

7.11 Donner le numéro et le nom des clients de Namur qui n'ont pas passé de commandes.

```
select NCLI, NOM
from   CLIENT
where  NCLI not in (select NCLI from COMMANDE)
and    LOCALITE = 'Namur'
      ou
select NCLI, NOM
from   CLIENT
where  not exists (select * from COMMANDE where NCLI = CLIENT.NCLI)
and    LOCALITE = 'Namur'
```

7.12 Quels sont les produits en sapin qui font l'objet d'une commande ?

```
select NPRO
from PRODUIT
where LIBELLE like '%SAPIN%'
and exists (select * from DETAIL where NPRO = PRODUIT.NPRO)
      ou
select NPRO
from PRODUIT
where LIBELLE like '%SAPIN%'
and NPRO in (select NPRO from DETAIL)
      ou
select distinct NPRO from PRODUIT P, DETAIL D (! jointure)
where LIBELLE like '%SAPIN%' and P.NPRO = D.NPRO;
```

7.13 Ecrire les requêtes SQL qui recherchent les clients (on simplifiera si nécessaire) :

- habitant à Lille ou à Namur.

```
select NCLI, LOCALITE
from   CLIENT
where LOCALITE in ('Lille','Namur')
```

- qui à la fois habitent à Lille et n'habitent pas à Namur.

= "qui habitent à Lille".

- qui habitent à Lille ou n'habitent pas à Namur.

= "qui n'habitent pas à Namur"

- qui n'habitent ni à Lille ni à Namur.

```
select NCLI, LOCALITE
from   CLIENT
where  LOCALITE not in ('Lille','Namur')
```

- qui n'habitent pas à Lille ou qui n'habitent pas à Namur.

= "tous les clients"

- de catégorie C1 habitant à Namur.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT = 'C1'
and    LOCALITE = 'Namur'
```

- de catégorie C1 ou habitant à Namur.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT = 'C1'
```

```
or     LOCALITE = 'Namur'
```

- de catégorie C1 n'habitant pas à Namur.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT = 'C1'
and    LOCALITE <> 'Namur'
```

- qui n'ont pas été sélectionnés dans la question précédente.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT <> 'C1'
or     CAT is null
or     LOCALITE = 'Namur'
```

- qui soit sont de catégorie B1 ou C1, soit habitent à Lille ou à Namur (ou les deux conditions).

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT in ('B1','C1')
or     LOCALITE in ('Lille','Namur')
```

- qui soit sont de catégorie B1 ou C1, soit habitent à Lille ou à Namur (mais pas les deux conditions).

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  (CAT in ('B1','C1') and LOCALITE not in ('Lille','Namur'))
or     (CAT not in ('B1','C1') and LOCALITE in ('Lille','Namur'))
```

- qui sont de catégorie B1 ou C1, et qui habitent à Lille ou à Namur.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT in ('B1','C1')
and    LOCALITE in ('Lille','Namur')
```

- qui n'ont pas été sélectionnés dans la question précédente.

```
select NCLI, LOCALITE, CAT
from   CLIENT
where  CAT not in ('B1','C1')
or     LOCALITE not in ('Lille','Namur')
```

c) Énoncés de type 3

7.14 Afficher la valeur totale des stocks (compte non tenu des commandes actuelles).

```
select sum(QSTOCK*PRIX) as TOTAL
from PRODUIT
```

7.15 Afficher le numéro et le libellé des produits en sapin :

- qui ne sont pas commandés,
- qui sont commandés à Toulouse,
- qui ne sont pas commandés à Toulouse
- qui ne sont commandés qu'à Toulouse,
- qui ne sont pas commandés qu'à Toulouse,
- qui sont commandés à Toulouse, mais aussi ailleurs.

7.16 Combien y a-t-il de commandes spécifiant un (ou plusieurs) produit(s) en acier ? *(! jointures)*

```
select count(*)
from COMMANDE
where  NCOM in (select NCOM
                from DETAIL
                where  NPRO in (select NPRO
                                from PRODUIT
                                where LIBELLE like '%ACIER%');
```

ou

```
select count(*)  (! jointures)
from   COMMANDE M
where  NCOM in (select NCOM
                from DETAIL D, PRODUIT P
                where  D.NPRO = P.NPRO
                and    LIBELLE like '%ACIER%')
```

ou

```
select count(distinct M.NCOM)  (! jointures)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
and    LIBELLE like '%ACIER%'
```

ou

7.17 Dans combien de localités trouve-t-on des clients de catégorie C1 ?

```
select count(distinct LOCALITE)
from   CLIENT
where  CAT = 'C1'
```

d) Énoncés de type 4

7.18 Exprimer de trois manières différentes la requête : *quels sont les produits qui ne sont pas commandés ?*

7.19 Afficher le numéro et le nom des clients qui n'ont pas commandé de produits en sapin.

```
select NCLI, NOM
from CLIENT
where NCLI not in (select NCLI from COMMANDE
                   where NCOM in (select NCOM from DETAIL
                                  where NPRO in (select NPRO
                                                 from PRODUIT
                                       where LIBELLE like '%SAPIN%')));
```

7.20 A la question : "*rechercher les localités dans lesquelles on n'a pas commandé de produit PA60*", quatre utilisateurs proposent les requêtes suivantes. Indiquer la (ou les) requêtes correctes, et interprétez les autres.

```
  select distinct LOCALITE
  from   CLIENT
  where  NCLI in
               (select NCLI
                from   COMMANDE
                where  NCOM in
                           (select NCOM
                            from   DETAIL
                            where  NPRO <> 'PA60'))
```

```
LOCALITE
====================
Lille
Poitiers
Toulouse
```

```
  select distinct LOCALITE
  from   CLIENT
  where  NCLI in
               (select NCLI
                from   COMMANDE
                where  NCOM not in
                           (select NCOM
                            from   DETAIL
                            where  NPRO = 'PA60'))
```

```
LOCALITE
====================
Lille
Poitiers
```

```
select distinct LOCALITE
from   CLIENT
where  NCLI not in
            (select NCLI
             from   COMMANDE
             where  NCOM in
                        (select NCOM
                         from   DETAIL
                         where  NPRO = 'PA60'))
```

```
LOCALITE
====================
Bruxelles
Geneve
Lille
Namur
Paris
Poitiers
Toulouse
```

```
select distinct LOCALITE
from   CLIENT
where  LOCALITE not in
            (select LOCALITE
             from   CLIENT
             where  NCLI in
                     (select NCLI
                      from   COMMANDE
                      where  NCOM in
                             (select NCOM
                              from   DETAIL
                              where  NPRO = 'PA60')))
```

seule expression correcte

```
LOCALITE
====================
Bruxelles
Geneve
Lille
Paris
```

7.21 Que signifie la requête suivante ?

```
select *
from   COMMANDE
where  NCOM not in (select NCOM
                    from   DETAIL
                    where  NPRO <> 'PA60')
```

7.22 Dans quelles localités a-t-on commandé en décembre 2008 ?

```
select distinct LOCALITE
from   CLIENT
where  NCLI in (select NCLI
                from   COMMANDE
                where  DATECOM like '%DEC-2008')
```

7.23 On suppose qu'on n'a pas trouvé utile de déclarer NCOM clé étrangère dans la table DETAIL. Il est donc possible que certaines lignes de DETAIL violent la contrainte d'intégrité référentielle portant sur cette colonne. Ecrire une requête qui recherche les anomalies éventuelles.

```
select NCOM, NPRO
from   DETAIL D
where  not exists (select * from COMMANDE where NCOM = D.NCOM);
```

7.24 Normalement, à toute commande doit être associé au moins un détail. Écrire une requête qui vérifie qu'il en bien ainsi dans la base de données.

```
select NCOM
from   COMMANDE M
where  not exists (select * from DETAIL where NCOM = M.NCOM);
```

7.25 Quels sont les produits (numéro et libellé) qui n'ont pas été commandés en 2008 ?

```
select NPRO, LIBELLE
from   PRODUIT
where  NPRO not in (select NPRO
                    from   DETAIL D
                    where  NCOM in (select NCOM
                                    from   COMMANDE M
                                    where  DATECOM like '%2008%');
```

ou

```
select NPRO, LIBELLE    (! jointure)
from   PRODUIT
where  NPRO not in (select NPRO
                    from   DETAIL D, COMMANDE M
                    where  D.NCOM = M.NCOM
                    and    M.DATECOM like '%2008%');
```

e) Énoncés de type 5

7.26 Rechercher les clients qui ont commandé tous les produits.

Suggestion. Application du quantificateur *pour tout.* On recherche les clients tels qu'il n'existe pas de produits qui n'apparaissent pas dans les détails de leurs commandes.

```
select NCLI
from CLIENT C
where not exists (select *
                  from PRODUIT
                  where NPRO not in
                              (select NPRO
                               from DETAIL
                               where  NCOM in (select NCOM
                                               from COMMANDE
                                                where NCLI = C.NCLI));

select NCLI    (! jointure)
from   CLIENT C
where not exists (select *
                  from PRODUIT
                  where NPRO not in (select NPRO
                                     from   DETAIL D, COMMANDE M
                                     where  D.NCOM = M.NCOM
                                     and    M.NCLI = C.NCLI))
```

7.27 Dans quelles localités a-t-on commandé tous les produits en acier (tous clients confondus) ?

Exercice préparatoire : *quels sont les clients qui ont commandé tous les produits en acier*.

```
select NCLI
from   CLIENT C
where not exists (select *
                  from   PRODUIT
                  where  LIBELLE like '%ACIER%'
                  and    NPRO not in (select NPRO
                                      from   DETAIL D, COMMANDE M
                                      where  D.NCOM = M.NCOM
                                      and    M.NCLI = C.NCLI))
```

Question d'origine :

```
les localités L telles
     qu'il n'existe pas
           de produits en acier qui ne soit commandé
                          par un client de la localité L

select distinct LOCALITE
from   CLIENT C
```

```
where not exists (select *
                  from   PRODUIT
                  where  LIBELLE like '%ACIER%'
                  and    NPRO not in
                            (select NPRO
                             from   DETAIL D, COMMANDE M, CLIENT CC
                             where  D.NCOM = M.NCOM
                             and    M.NCLI = C.NCLI
                             and    CC.LOCALITE = C.LOCALITE))
```

7.28 Rechercher les produits qui ont été commandés par tous les clients.

```
select NPRO
from   PRODUIT P
where  not exists (select *
                   from   CLIENT
                   where  NCLI not in (select NCLI
                                       from   COMMANDE M, DETAIL D
                                       where  M.NCOM = D.NCOM
                                       and    D.NPRO = P.NPRO))
```

7.29 Rechercher les localités dont aucun client n'a passé de commande.

```
select distinct LOCALITE
from   CLIENT
where  LOCALITE not in (select LOCALITE
                        from   CLIENT C
                        where exists (select *
                                      from   COMMANDE
                                      where  NCLI = C.NCLI))
```

7.30 Rechercher les localités dont tous les clients ont passé au moins une commande.

```
select distinct LOCALITE
from   CLIENT C
where  not exists (select * from CLIENT
                   where  LOCALITE = C.LOCALITE
                   and    NCLI not in (select NCLI from COMMANDE))
```

7.31 Rechercher les produits qui sont commandés dans toutes les localités.

```
select NPRO
from   PRODUIT P
where not exists
           (select *
            from   CLIENT
            where  LOCALITE not in
                      (select LOCALITE
                       from   CLIENT C, COMMANDE M, DETAIL D
```

```
where  M.NCLI = C.NCLI
and    D.NCOM = M.NCOM
and    D.NPRO = P.NPRO))
```

A.8 CHAPITRE 8 - LE LANGAGE SQL DML (2)

a) Énoncés de type 2

8.1 Calculer le montant de chaque détail de commande du client 'C400'.

```
select NCOM, P.NPRO, QCOM*PRIX as MONTANT
from   DETAIL D, PRODUIT P
where  D.NPRO = P.NPRO and NCLI = 'C400';
```

8.2 Calculer le montant commandé des produits en sapin.

```
select sum(QCOM*PRIX) as MONTANT
from   DETAIL D,PRODUIT P
where  D.NPRO = P.NPRO
and    P.LIBELLE like '%SAPIN%';
```

b) Énoncés de type 3

8.3 Afficher le total et la moyenne des comptes des clients, ainsi que le nombre de clients, selon chacune des classifications suivantes :

- par catégorie,

```
select CAT, sum(COMPTE), avg(COMPTE), count(*)
from   CLIENT group  by CAT;
```

- par localité,

- par catégorie dans chaque localité.

```
select CAT, sum(COMPTE), avg(COMPTE), count(*)
from   CLIENT group  by LOCALITE, CAT;
```

8.4 Combien y a-t-il de commandes spécifiant un (ou plusieurs) produit(s) en acier (on proposera une solution différente de celle du chapitre 7) ?

```
select count(distinct M.NCOM)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
and    LIBELLE like '%ACIER%';
     ou
select count(*)
from   COMMANDE M
where  NCOM in (select NCOM
                from DETAIL D, PRODUIT P
                where  D.NPRO = P.NPRO
                and    LIBELLE like '%ACIER%');
```

8.5 Créer une table et y ranger les données suivantes relatives aux détails de commande : numéro et date de la commande, quantité commandée, numéro et prix du produit, montant du détail.

```
create table (NCOM ..., DATECOM ..., ..., MONTANT ...);

insert into DETAIL_COM(NCOM, DATECOM, QCOM, NPRO, PRIX, MONTANT)
select M.NCOM, DATECOM, QCOM, P.NPRO, PRIX, QCOM*PRIX
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO;
```

8.6 Annuler les comptes négatifs des clients de catégorie C1.

```
update CLIENT
set    COMPTE = 0
where  COMPTE < 0
and    CAT = 'C1';
```

8.7 Compléter le fragment suivant de manière à former une requête valide
`select CAT, NPRO, sum(QCOM*PRIX) from ...`

```
select CAT, NPRO, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by CAT, P.NPRO;
```

8.8 En se basant sur le schéma 8.5, écrire les requêtes SQL qui donnent :

- les matières premières (produits qui n'ont pas de composants);

```
select NPRO
from   PRODUIT
where  NPRO not in (select COMPOSE from COMPOSITION);
```

- les produits finis (qui n'entrent dans la composition d'aucun autre);

```
select NPRO
from   PRODUIT
where  NPRO not in (select COMPOSANT from COMPOSITION);
```

- les produits semi-finis (tous les autres);

```
select NPRO
from   PRODUIT
where  NPRO in (select COMPOSE   from COMPOSITION)
```

```
and    NPRO in (select COMPOSANT from COMPOSITION);
```

- le prix et poids unitaires d'un produit fini ou semi-fini dont tous les composants ont un poids et un prix unitaires.

```
select PH.NPRO, sum(QTE*PB.PRIX_U), sum(QTE*PB.POIDS_U)
from   PRODUIT PH, COMPOSITION C, PRODUIT PB
where  PH.NPRO = C.COMPOSE
and    C.COMPOSANT = PB.NPRO
and    not exists (select *
                   from  COMPOSITION CC, PRODUIT BB
                   where CC.COMPOSANT = BB.NPRO
                   and   CC.COMPOSE = PH.NPRO
                   and   (BB.PRIX_U is null
                          or BB.POIDS_U is null))
group by PH.NPRO;
```

8.9 Quels sont les personnes qui ont le même responsable que p4 ?

```
select NPERS, NOM
from   PERSONNE
where  RESPONSABLE in (select RESPONSABLE from PERSONNE
                       where NPERS = 'p4';
```

c) Énoncés de type 4

8.10 Calculer le montant de chaque commande.

8.11 Calculer le montant dû par chaque client. Dans ce calcul, on ne prend en compte que le montant des commandes. Attention aux clients qui n'ont pas passé de commandes.

```
select NCLI, sum(QCOM*PRIX)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by NCLI
union
select NCLI, 0
from   CLIENT C
where  not exists (select * from COMMANDE where NCLI = C.NCLI);
```

8.12 Calculer le montant dû par les clients de chaque localité. Dans ce calcul, on ne prend en compte que le montant des commandes. Attention aux localités dans lesquelles aucun client n'a passé de commandes.

```
select LOCALITE, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
```

```
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE
union
select distinct LOCALITE, 0
from   CLIENT
where  LOCALITE not in (select LOCALITE
                        from   CLIENT C
                        where  exists (select *
                                       from   COMMANDE
                                       where  NCLI = C.NCLI));
```

8.13 Calculer, par jour, le total des montants des commandes.

```
select DATECOM, sum(QCOM*PRIX)
from   COMMANDE M, DETAIL D, PRODUIT P
where  M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by DATECOM;
```

8.14 On suppose qu'on n'a pas trouvé utile d'imposer un identifiant primaire sur la table PRODUIT. Il se peut donc que plusieurs lignes aient même valeur de la colonne NPRO, ce qui viole le principe d'unicité des valeurs de cette colonne.

- Ecrire une requête qui recherche les valeurs de NPRO présentes en plus d'un exemplaire.

```
select NPRO, count(*)
from   PRODUIT
group by NPRO
having count(*) > 1
```

- Écrire une requête qui indique combien de valeurs de NPRO sont présentes en plus d'un exemplaire.

```
create view OCCURRENCES(NPRO,NOMBRE) as
   select NPRO, count(*)
   from   PRODUIT
   group by NPRO;

select count(*)
from   OCCURRENCES
where  NOMBRE > 1;
```

- Écrire une requête qui indique combien de lignes comportent une erreur d'unicité de NPRO.

- Écrire une requête qui, pour chaque valeur de NPRO présente dans la table, indique dans combien de lignes cette valeur est présente.

- Écrire une requête qui, pour chaque valeur de NPRO qui n'est pas unique, indique dans combien de lignes cette valeur est présente.

- Écrire une suite de requêtes qui crée une table contenant les numéros de NPRO qui ne sont pas uniques.

8.15 Afficher pour chaque localité, les libellés des produits qui y sont commandés.

```
select LOCALITE, LIBELLE
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, LIBELLE
order by LOCALITE, LIBELLE
```

8.16 Afficher par localité, et pour chaque catégorie dans celle-ci, le total des montants des commandes.

```
select LOCALITE, CAT, sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, CAT
   union
select LOCALITE, CAT, 0
from   CLIENT C
group by LOCALITE, CAT
having not exists (select *
                   from COMMANDE
                   where  NCLI in (select NCLI
                                   from   CLIENT
                                   where  LOCALITE = C.LOCALITE
                                   and    CAT = C.CAT));
```

8.17 Indiquer, pour chaque localité, les catégories de clients qui n'y sont pas représentées

Suggestion. Construire l'ensemble de tous les couples (LOCALITE, CAT) possibles et en retirer ceux qui existent dans la base. Attention aux valeurs `null`, qui ne doivent pas être prises en compte.

```
select distinct L.LOCALITE, C.CAT
from   CLIENT L, CLIENT C
where  CAT is not null
and not exists (select *
                from CLIENT
                where LOCALITE = L.LOCALITE and CAT = C.CAT);
```

8.18 Produire (à l'écran) une table de couples <X,Y> de clients tels que X et Y habitent dans la même localité. On évitera de renseigner <X,X>, mais aussi <Y,X> si <X,Y> est déjà repris.

Suggestion. Auto-jointure de CLIENT. On évitera les couples inverse en imposant un ordre sur les valeurs de NPRO (p.ex. X < Y).

```
select C1.LOCALITE, C1.NCLI, C2.NCLI
from   CLIENT C1, CLIENT C2
where  C1.NCLI < C2.NCLI
and    C1.LOCALITE = C2.LOCALITE
order by C1.LOCALITE, C1.NCLI, C2.NCLI
```

8.19 En se basant sur le schéma 8.5, écrire une requête de mise à jour qui complète les prix et poids unitaires des produits finis ou semi-finis. Pour simplifier la procédure, on admet que cette requête soit exécutée autant de fois que nécessaire pour que tous les produits soient complétés.

```
update PRODUIT PH
set PRIX_U = (select  sum(QTE*PB.PRIX_U)
              from    COMPOSITION C, PRODUIT PB
              where   PH.NPRO = C.COMPOSE
              and     C.COMPOSANT = PB.NPRO
              group by PH.NPRO)
where  PRIX_U is null
and    not exists (select * from  COMPOSITION CC, PRODUIT BB
                   where CC.COMPOSANT = BB.NPRO
                   and   CC.COMPOSE = PH.NPRO
                   and   BB.PRIX_U is null);
```

8.20 En considérant le schéma de la figure 8.7, calculer pour chaque ville, le prix moyen de chaque produit.

```
select VILLE, PRODUIT, avg(PRIX)
from   VENTE V, LOCALISATION L
where  V.CHAINE = L.CHAINE
group by VILLE, PRODUIT;
```

8.21 Afficher pour chaque client, le nombre de commandes, le nombre de produits commandés et le nombre de détails. On se limite aux clients qui ont passé au moins une commande.

Suggestion : il s'agit d'une requête basée sur des groupements multi-niveaux.

```
select  C.NCLI, count(distinct NCOM) as Commandes,
        count(distinct NPRO) as Produits,
        count(NPRO) as Details
from    CLIENT C, COMMANDE M, DETAIL D
where   C.NCLI = M.NCLI and M.NCOM = D.NCOM
group by C.NCLI
```

```
select  C.NCLI, count(distinct NCOM) as Commandes,
        count(distinct NPRO) as Produits,
        count(NPRO) as Details
from    CLIENT C, COMMANDE M, DETAIL D
where   C.NCLI = M.NCLI and M.NCOM = D.NCOM
group by C.NCLI;
```

8.22 Afficher, pour chaque localité et pour chaque catégorie, (1) le nombre de commandes passées par les clients de cette localité et de cette catégorie, (2) le montant total de ces commandes.

```
select LOCALITE, CAT, count(distinct NCOM), sum(QCOM*PRIX)
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
group by LOCALITE, CAT
```

8.23 On considères trois tables T1, T2 et T3, chacune possédant un unique attribut A identifiant. Calculer l'ensemble des valeurs de A qui

- sont présentes dans les trois tables,

```
select A from T1
intersect
select A from T2
intersect
select A from T3;
     ou
select A from T1
where A in (select A from T2)
and A in select A from T3);
```

- sont présentes dans T1 seulement,

```
select A from T1
except
select A from T2
except
select A from T3;
     ou
select A from T1
where A not in (select A from T2)
and A not in select A from T3);
```

- sont présentes dans T1 et T2 mais pas dans T3,

```
select A from T1
intersect
select A from T2
except
```

```
select A from T3;
     ou
select A from T1
where A in (select A from T2)
and A not in select A from T3);
```

- sont présentes dans une seule table,

```
create view T123(A ...)
as select A from T1
union all select A from T2
union all select A from T2;

select A from T123 group by A having count(*) = 1;
```

- sont présentes dans deux tables seulement.

```
select A from T123 group by A having count(*) = 3;
```

d) Énoncés de type 5

8.24 Calculer le nombre moyen de produits par commande. De même : le nombre moyen de commandes par client, par localité ou par catégorie.

Remarque. Il n'est pas possible de demander directement la moyenne d'une somme. *Suggestion* : construction et interrogation d'une vue ou calcul de la moyenne dans le `from`.

```
create view NOMBRE(NCOM,NBRE) as
select NCOM, count(*)
from   DETAIL
group by NCOM;

select avg(NBRE)
from NOMBRE;
```

8.25 Quel est, pour chaque localité, le nombre moyen de commandes par client.

```
create view NOMBRE(NCLI,NBRE) as
select C.NCLI, count(*)
from   CLIENT C, COMMANDE M
where  C.NCLI = M.NCLI
group by C.NCLI;

select LOCALITE, avg(NBRE)
from   CLIENT C, NOMBRE N
where  C.NCLI = N.NCLI
group by LOCALITE;
```

8.26 Ecrire une requête SQL qui donne, pour chaque catégorie de produit, le nombre de produits qui ont été commandés le 23-12-2008.

```
select CAT, count(distinct NPRO)
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
AND    DATECOM = '23-DEC-2008'
group by CAT
```

8.27 Donner pour chaque localité dans laquelle se trouve au moins un client de catégorie 'C1' la liste des produits en sapin qu'on y a commandés.

```
select LOCALITE, D.NPRO
from   CLIENT C, COMMANDE M, DETAIL D, PRODUIT P
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO = P.NPRO
and    LOCALITE in (select LOCALITE from CLIENT where CAT = 'C1')
and LIBELLE like '%SAPIN%'
group by LOCALITE, D.NPRO
```

8.28 Donner pour chaque produit la liste des localités dans lesquelles ce produit est commandé en plus de 500 unités (= au total pour la localité).

```
select D.NPRO, LOCALITE, sum(QCOM)
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
group by D.NPRO, LOCALITE
having sum(QCOM) > 500
```

8.29 Afficher, pour chaque localité, les produits qu'on y commande et qui sont aussi commandés dans au moins une autre localité.
Suggestion. Un produit est intéressant si le nombre de localités dans lesquelles il est commandé est supérieur ou égal à 2.

```
select LOCALITE, NPRO
from   CLIENT C, COMMANDE M, DETAIL D
where  C.NCLI = M.NCLI
and    M.NCOM = D.NCOM
and    D.NPRO in (select NPRO
                  from   CLIENT C, COMMANDE M, DETAIL D
                  where  C.NCLI = M.NCLI
                  and    M.NCOM = D.NCOM
                  group by NPRO
                  having count(distinct LOCALITE) > 1)
group by LOCALITE, D.NPRO
```

8.30 Dans quelles localités peut-on trouver au moins un client qui a commandé tous les produits en sapin, et ce, pour un montant total, pour ces produits, dépassant 10.000 ?

8.31 Calculer, pour chaque localité, le nombre de catégories distinctes.

```
select LOCALITE, count(distinct CAT)
from   CLIENT
where  CAT is not null
group by LOCALITE
```

8.32 La section 8.10.1 suggère une comparaison entre un instantané et une vue de même définition. En effet, ces deux techniques pourraient être considérées comme équivalentes pour la formulation de requêtes. Établir une liste de critères de comparaison et l'appliquer aux deux techniques.

8.33 Mettre à jour les comptes des clients en en déduisant le montant des commandes en cours.

Suggestion. *cfr* mise à jour des quantités en stock des produits; attention aux clients qui n'ont pas commandé.

```
update PRODUIT P
set QSTOCK = QSTOCK - (select sum(QCOM)
                       from   DETAIL
                       where  NPRO = P.NPRO)
where exists (select *
              from   DETAIL
              where  NPRO = P.NPRO)
```

8.34 Mettre à jour les quantités en stock des produits en en déduisant les quantités commandées par les clients de catégorie B1 et C1.

```
update PRODUIT P
set QSTOCK = QSTOCK - (select sum(QCOM)
                       from   DETAIL D, COMMANDE M, CLIENT C
                       where  D.NPRO = P.NPRO
                       and    D.NCOM = M.NCOM
                       and    M.NCLI = C.NCLI
                       and    CAT in ('B1','C1'))
where exists (select *
              from   DETAIL D, COMMANDE M, CLIENT C
              where  D.NPRO = P.NPRO
              and    D.NCOM = M.NCOM
              and    M.NCLI = C.NCLI
              and    CAT in ('B1','C1'))
```

8.35 On suppose qu'on dispose de deux tables PRODUIT1 et PRODUIT2 décrivant des produits de deux filiales distinctes, et de même structure que PRODUIT. Il est possible qu'un produit soit présent simultanément dans les deux filiales. Le même numéro de produit est repris alors dans les deux tables. On désire intégrer les deux dépôts. En conséquence, on fusionne les deux tables initiales en une troisième selon les règles suivantes :

- si un produit n'est présent que dans une seule filiale, sa ligne est reprise telle quelle,
- si un produit est présent dans les deux filiales, on ne le décrit qu'une seule fois. Son libellé est celui de PRODUIT1, son prix est le minimum des deux valeurs et sa quantité en stock est la somme des deux valeurs.

Ecrire les requêtes d'intégration.

```
create table PRODUIT1
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT1 values ('A001','Farine', 0.8, 65);
insert into PRODUIT1 values ('B001','Sel', 0.5, 120);
insert into PRODUIT1 values ('A003','Huile olive', 5.2, 9);

create table PRODUIT2
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT2 values ('A002','Sucre', 1.2, 45);
insert into PRODUIT2 values ('B001','Sel iodé', 0.45, 35);
insert into PRODUIT2 values ('A003','Huile', 5.6, 22);

create table PRODUIT3
(NPRO char(10) not null primary key,
 LIBELLE char(30) not null,
 PRIX decimal(5,2) not null,
 QSTOCK decimal(6) not null);

insert into PRODUIT3
select *
from PRODUIT1 P1
where not exists (select * from PRODUIT2 where NPRO = P1.NPRO);

insert into PRODUIT3
select *
from PRODUIT2 P2
where not exists (select * from PRODUIT1 where NPRO = P2.NPRO);

insert into PRODUIT3
select P1.NPRO, P1.LIBELLE, P1.PRIX, P1.QSTOCK + P2.QSTOCK
from   PRODUIT1 P1, PRODUIT2 P2
```

```
where  P1.NPRO = P2.NPRO
and    P1.PRIX <= P2.PRIX;

insert into PRODUIT3
select P1.NPRO, P1.LIBELLE, P2.PRIX, P1.QSTOCK + P2.QSTOCK
from   PRODUIT1 P1, PRODUIT2 P2
where  P1.NPRO = P2.NPRO
and    P1.PRIX > P2.PRIX;
```

8.36 Soit une jointure de deux ou plusieurs tables, dont la clause `select` spécifie certaines colonnes. On désire que le résultat ne comporte pas de lignes en double. Indiquer dans quelles conditions le modifieur `distinct` est inutile.

Suggestion. `distinct` est inutile lorsque la clause `select` comprend tous les composants de l'identifiant de la table mentionnée dans la clause `from` ou (cas d'une jointure par exemple) construite dans les clauses `from-where-group-by`.

8.37 On considère une base de données constituée des deux tables suivantes : R(A,B) et S(B,C). Evaluer l'ordre de grandeur du nombre de requêtes différentes qu'il est possible de formuler sur cette base de données.

Réponse : une infinité.

e) Énoncés de type 6

8.38 La figure 11.27 (Voyages en train) représente un schéma qui est accompagné de suggestions de questions auxquelles une base de données traduisant ce schéma permettrait de répondre. On suppose que ce schéma est traduit en structure de tables (*cf.* exercice 14.3). Exprimer chacune de ces questions sous la forme de requêtes SQL.

8.39 On désire réaliser le couplage deux à deux de certaines localités où résident des clients. À cette fin, on désire construire des listes de couples candidats. On se limite cependant aux couples de localités dans lesquelles on achète au moins un même produit.

8.40 Même question que 8.39 ci-dessus, mais on se limite aux couples de localités dans lesquelles on achète exactement les mêmes produits.

8.41 Pour chaque produit, indiquer la ville dans laquelle la quantité totale commandée est minimale, et celle dans laquelle cette quantité est maximale.

f) Énoncé de type 7

8.42 Rédiger un script (suite de requêtes) SQL qui réalise en fin de mois les opérations de gestion régies par les règles suivantes :

- pour chaque client, on calcule le total des montants des commandes du mois écoulé (= les commandes d'un mois et d'une année déterminés);
- si ce montant est supérieur à celui du mois précédent, on applique une réduction de 5%;
- ensuite, si le client est le seul de sa localité, on applique une réduction supplémentaire de 5%;
- on range dans une table les informations nécessaires à l'édition des factures du mois;
- on met à jour le compte des clients;
- on met à jour le niveau de stock (QSTOCK) des produits commandés;
- pour chaque produit dont le niveau de stock a été mis à jour, et dont le niveau est inférieur ou égal à 0, on prépare dans une table adéquate les informations de rapprovisionnement.

A.9 CHAPITRE 9 - LE LANGAGE SQL AVANCÉ

9.1 Développer, à l'aide des concepts étudiés dans cet ouvrage, un mécanisme qui permet de limiter l'accès à une table par un utilisateur déterminé à des moments bien déterminés, par exemple du lundi au vendredi, de 9h à 17h.

Solution

On suggère de définir une vue limitant l'accès aux plages horaires désirées.

```
create view T_CLIENT(NCLI, NOM, ...) as
select NCLI, NOM, ...
from CLIENT
where extract(weekday from current_date) between 1 and 5
and   extract(hour from current_time) between 9 and 17;
revoke select on CLIENT from U_MERCIER:
grand select on T_CLIENT to U_MERCIER;
```

9.2 Exprimer l'opérateur `except` à l'aide d'une jointure externe.

Solution

Exemple : liste des numéros des clients qui n'ont pas passé de commande :

```
select C.NCLI
from CLIENT C left outer join COMMANDE M
     on (C.NCLI = M.NCLI)
where NCOM is null;
```

9.3 Ecrire une requête qui produise les supérieurs hiérarchiques, directs ou indirects, d'une personne dont on spécifie le numéro (figure 8.2).

9.4 Ecrire une requête qui donne la liste des produits semi-finis et matières premières qui entrent dans la composition du produit de numéro `p1` (figure 8.6).

9.5 Ecrire une requête qui donne la liste des matières premières qui entrent dans la composition du produit de numéro `p1` (figure 8.6).

9.6 Ecrire une requête qui calcule le prix d'un produit de numéro spécifié (figure 8.6).

9.7 Ecrire une requête qui donne, pour chaque table, le nombre de colonnes et la longueur maximum des lignes.

9.8 Le catalogue que nous avons décrit (figures 10.3 et 10.4) diffère quelque peu des catalogues proposés par les SGBD. En particulier, les catalogues réels ont souvent une structure complexe destinée à améliorer les performances. Ecrire les requêtes qui garnissent les tables de la figure 10.4 à partir du

contenu des tables du catalogue de votre SGBD. Pour fixer les idées, on pourra s'exercer sur la structure ci-dessous, limitée à la représentation des *clés* (identifiants et clés étrangères) :

SYS_KEY_COL					
TNAME	**KTYPE**	**KEYID**	**CNAME**	**TARGTAB**	**TARGCOL**
CLIENT	P	K1	NCLI		
COMMANDE	P	K2	NCOM		
COMMANDE	F	K3	NCLI	CLIENT	NCLI
PRODUIT	P	K7	NPRO		
DETAIL	P	K4	NCOM		
DETAIL	P	K4	NPRO		
DETAIL	F	K5	NCOM	COMMANDE	NCOM
DETAIL	F	K6	NPRO	PRODUIT	NPRO

Une ligne de cette table représente un composant d'une *clé*. Elle indique successivement le nom de la table de la *clé*, le type de la clé (*P*rimary, *F*oreign), son identifiant interne et le nom de la colonne; s'il s'agit d'une clé étrangère, elle indique en outre le nom de la table et de la colonne cibles.

9.9 Le lecteur attentif aura observé que la table SYS_KEY_COL de l'exercice 9.8 n'est pas normalisée. On l'invite à repérer les dépendances fonctionnelles anormales puis à décomposer la table de manière à obtenir des fragments normalisés.

9.10 Ecrire une requête qui met à jour la valeur de COMPTE de chaque client en en retirant le total des montants des commandes de la date courante.

9.11 Proposer un schéma de tables équivalent à celui de la figure 2.7 établi selon concepts relationnels-objet. On remplacera les clés étrangères par des colonnes de référence et on représentera la table DETAIL par une colonne complexe (tableau de 20 rows) de la table COMMANDE.

9.12 La requête ci-dessous est tirée d'une application Access fournie par Microsoft dans la distribution d'Office. Les noms de tables et de colonnes ont été anonymisés mais la structure a été conservée. Réécrivez cette requête en exprimant les jointures sous une forme standard.

```
select distinct *
from S inner join (P inner join ((E inner join
(C inner join A on C.I = A.CI) on E.I = A.EI)
inner join B on A.I = B.OI) on P.I = B.PI) on S.I = A.S;
```

9.13 Au moment de son entrée en fonction, les utilisateurs Jean et Andre ne jouissent d'aucun privilège sur la base de donnée de son entreprise. Les utilisateurs U1, U2 et Jean exécutent ensuite les requêtes (valides) de gestion de privilèges suivantes, dans cet ordre :

```
U1:   grant select, update(QSTOCK) on PRODUIT to Jean
      with grand option;
U2:   grant select, update(QSTOCK,PRIX) on PRODUIT to Jean
      with grant option;
Jean: grant select, update(PRIX) to André;
U1:   revoke grant option for update(QSTOCK) on PRODUIT
      from Jean;
U2:   revoke update(QSTOCK) on PRODUIT from Jean;
```

Indiquer les privilèges des utilisateurs Jean et Andre après l'exécution de chaque requête de cette séquence. Poser les hypothèses additionnelles nécessaires.

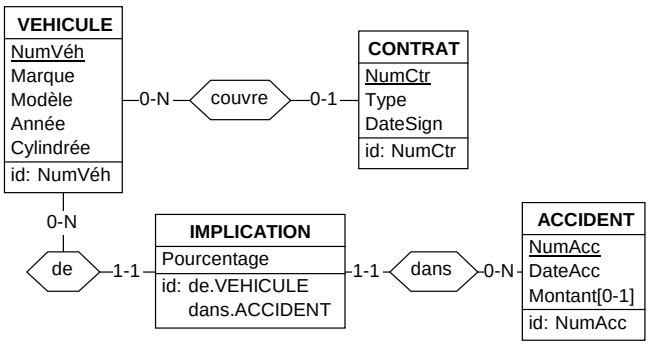
A.10 CHAPITRE 10 - CONSTRUCTION D'UNE BASE DE DONNÉES

Néant

A.11 CHAPITRE 11 - LE MODÈLE ENTITÉ-ASSOCIATION DE BASE

11.1 On considère le schéma de la figure 11.18. Admettons à présent que, toutes choses restant égales par ailleurs, (1) un véhicule puisse être couvert par un nombre quelconque de contrats, (2) on désire connaître, chaque fois qu'un véhicule est impliqué dans un accident, le pourcentage de responsabilité qui lui a été attribué. Modifier le schéma en conséquence.

Solution

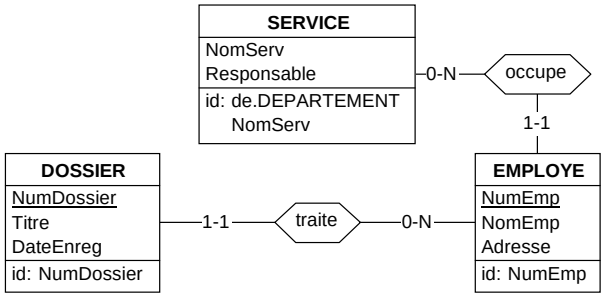


11.2 En utilisant les conventions graphiques des figures 11.6 et 11.16, dessiner les populations (entités, associations, valeurs d'attributs) correspondant au contenu de la base de données de la figure 2.8.

11.3 En utilisant le dessin élaboré lors de la résolution de l'exercice 11.2, retrouver les informations suivantes, selon le procédé de la section 11.8 :
– quels sont les produits commandés à Poitiers ?
– quels clients ont commandé le produit PA45 le 2/1/2009 ?

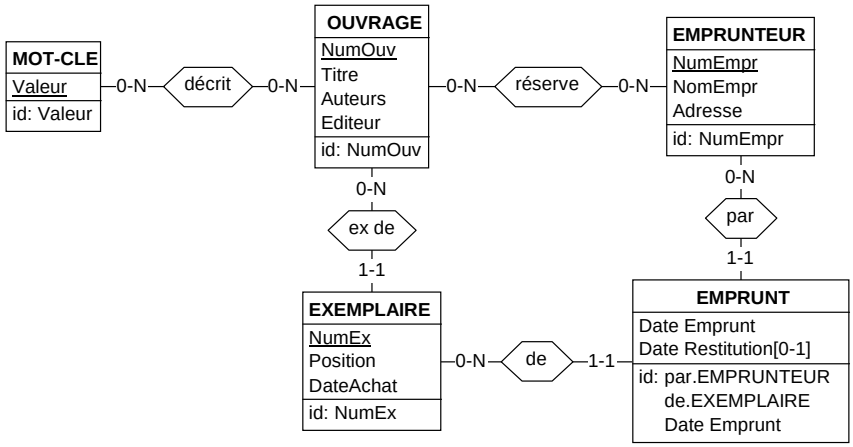
11.4 Dans l'interprétation du schéma 11.25, on précise qu'un service est réputé traiter un dossier dès qu'un de ses employés est en charge de ce dossier. Modifier le schéma pour tenir compte de cette précision.

Solution



11.5 Modifier le schéma de la figure 11.26 de manière à y représenter
a) la réservation d'un ouvrage (ou d'un exemplaire ?) par un emprunteur;
b) l'historique des emprunts d'ouvrages (ou d'exemplaires ?).

Solution



11.6 Dessiner un graphe de populations représentatives pour le schéma de la figure 11.27. On y représentera une situation réelle, qu'on trouvera dans un horaire de chemin de fer.

11.7 Pourquoi dans le schéma de la figure 11.27 un agent conduit-il un voyage et non un train, comme il est d'usage ?

11.8 Le schéma de la figure 11.27 permet-il de résoudre le problème suivant ? Lorsqu'un conducteur X désire transmettre un colis à un autre conducteur Y, il dépose ce colis à une station par laquelle il passe, et par laquelle Y passera plus tard. X et Y se posent la question suivante à la date Z : où (quelle station) et à quelle date au plus tôt, Y pourra-t-il prendre livraison d'un colis que X veut lui transmettre. On suppose qu'un voyage ne s'étend pas sur plus d'une journée, et qu'un colis déposé à une date déterminée pourra être enlevé dès le lendemain.

11.9 Pourquoi dans le schéma de la figure 11.27 les sections ne sont-elles pas identifiées par leurs stations de départ et d'arrivée ?

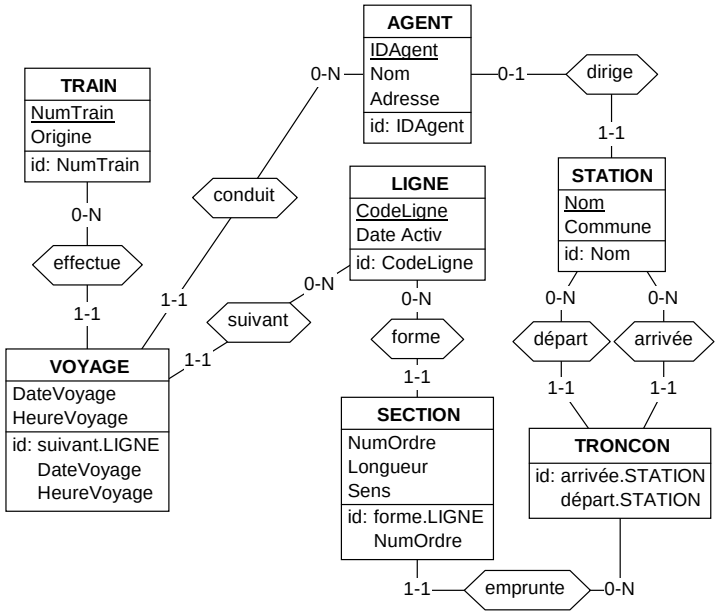
Solution

Parce qu'une section est un fragment de ligne et non la portion de voie qui relie deux stations. Il existe autant de sections entre deux stations qu'il y a de lignes passant consécutivement par ces stations.

11.10 Le schéma de la figure 11.27 représente des sections de ligne de telle manière que deux lignes empruntant le même tronçon de voie définissent deux sections différentes. Donner au concept de *tronçon de voie* une définition précise et modifier le schéma conceptuel de manière qu'il représente non seulement les sections mais également les tronçons de voie.

Solution

Un **tronçon** est une liaison directe, ininterrompue, entre deux stations; il n'existe pas de station au milieu d'un tronçon. Conventionnemment, on fixe pour un tronçon une station de **départ** et une station d'**arrivée**. Il n'existe pas plus d'un tronçon entre deux stations, bien qu'un tronçon puisse comporter plusieurs portions de voies physiques (= rails). Une section d'une ligne emprunte un tronçon dans le **sens** *direct* (départ → arrivée) ou dans le sens *inverse* (arrivée → départ).



11.11 Toujours relativement à ce même schéma, affiner le concept de *train*, en considérant que celui-ci est constitué de motrices, de voitures de voyageur, de wagons de marchandise, de fourgons, *etc.*, dans un ordre déterminé.

11.12 On considère les deux types d'associations r entre A et B et s entre B et C. On considère aussi le type d'associations rs défini comme la composition de r et s (figure 1.2). Déterminer la classe fonctionnelle de rs à partir de celles de r et de s. Cette question est très importante pour comprendre l'information contenue dans un schéma. En considérant le schéma de la figure 11.18, elle pourrait par exemple se concrétiser comme suit :

- combien de véhicules sont couverts par les contrats signés par un client ?
- combien de contrats couvrent les véhicules appartenant à un client ?
- combien de clients sont propriétaires des véhicules couverts par un contrat ?
- combien de propriétaires sont impliqués dans un accident ?
- combien de clients ont signé les contrats couvrant les véhicules d'un propriétaire (double composition) ?

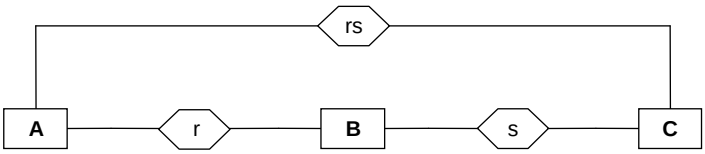


Figure 1.2 - Etude de la composition de deux types d'associations

Solution

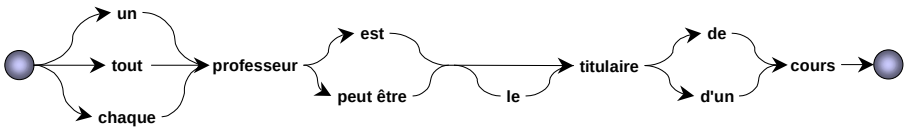
Pour simplifier le raisonnement, on considère les quatre classes fonctionnelles 1:N, N:1, 1:1 et N:N (N:1 est simplement la classe fonctionelle d'un type d'association 1:N considéré dans le sens inverse). On définit ainsi 16 configurations distinctes.

r	s	rs
1:N	1:N	1:N
1:N	N:1	N:N
1:N	1:1	1:N
1:N	N:N	N:N
N:1	1:N	N:N
N:1	N:1	N:1
N:1	1:1	N:1
N:1	N:N	N:N
1:1	1:N	1:N
1:1	N:1	N:1
1:1	1:1	1:1
1:1	N:N	N:N
N:N	1:N	N:N
N:N	N:1	N:N
N:N	1:1	N:N
N:N	N:N	N:N

A.12 CHAPITRE 12 - ÉLABORATION D'UN SCHÉMA CONCEPTUEL

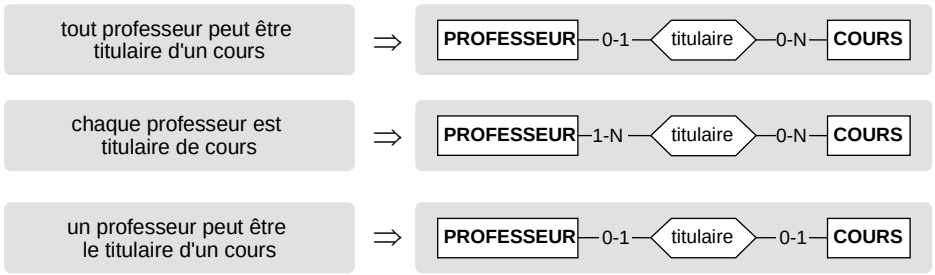
12.1 On considère le fragment de schéma :

ainsi que le graphe ci-dessous, dont chacun des 24 chemins obtenus en suivant les flèches de gauche à droite génère une proposition binaire conforme à ce schéma. Par exemple : *tout professeur peut être titulaire d'un cours*.



Pour chacune de ces propositions, compléter le schéma en y ajoutant les contraintes de cardinalité qu'elle suggère.

Solution



12.2 Décomposer le fragment de texte ci-dessous en propositions élémentaires (attention aux propositions irréductibles) :

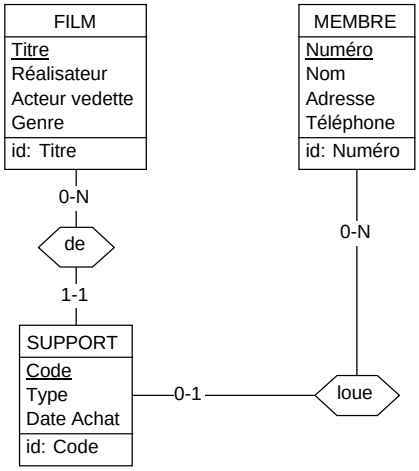
On considère des entreprises manufacturières nationales. Chacune est identifiée par son code et possède une dénomination, une adresse et un numéro de TVA qui lui est propre. Une adresse est constituée d'un nom de voie, d'un nom de localité et d'un code postal. Les entreprises exportent des produits vers des pays étrangers. Une entreprise affecte un délégué commercial à chaque pays dans lequel elle exporte.

12.3 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

Un club vidéo propose des cassettes et des DVD en location à ses membres. Pour chaque membre, on enregistre le nom, l'adresse, le numéro de téléphone. On lui donne un numéro d'inscription qui l'identifie. Chaque support est caractérisé par son type (cassette ou DVD), un code identifiant et la date d'achat. Pour le film du support, on enregistre le titre (identifiant),

son réalisateur, l'acteur vedette et le genre. Plusieurs supports peuvent être disponibles pour un même film, alors que pour certains films, il n'existe pas encore de supports proposés à la location. A tout instant, un support peut être loué par un membre du club.

Solution

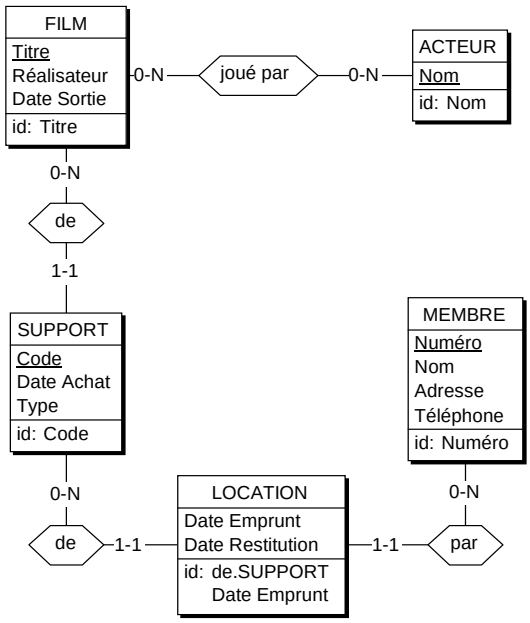


12.4 L'énoncé qui suit complète celui du premier exercice :

Chaque support est en outre caractérisé par le nombre d'emprunts, le titre du film, son réalisateur, ses principaux acteurs (nom et prénom, tous deux identifiants, ainsi que la date de naissance), la date de sortie du film. Pour chaque location, on connaît le support, la date d'emprunt, la date normale de restitution, la date de restitution effective et l'emprunteur. Une location dure un nombre entier de jours (au moins un). On conserve l'historique des locations.

Suggestion. On représentera les emprunts en cours et les emprunts clôturés par un même objet.

Solution

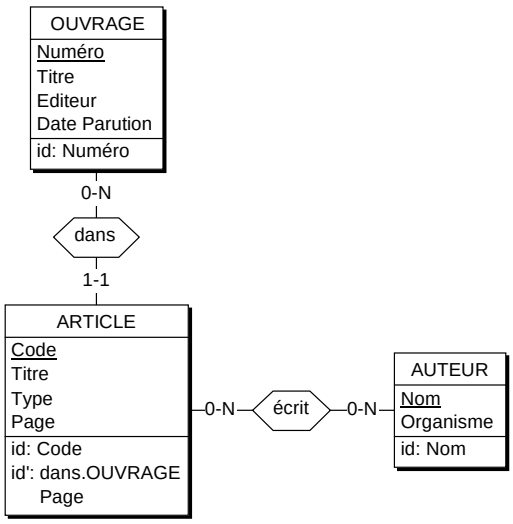


12.5 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

On désire gérer une bibliographie constituée d'articles (code identifiant, type, titre). Chaque article est écrit par un nombre quelconque d'auteurs. Chaque auteur est caractérisé par son nom (supposé identifiant) et l'organisme dont il dépend. En outre, chaque article est extrait d'un ouvrage dont on donne le titre, l'éditeur, la date de parution, ainsi qu'un numéro identifiant. Dans cet ouvrage, chaque article commence en haut d'une page dont on connaît le numéro. ***Exemple*** *: l'article de code 13245, du type «THEORIE», intitulé «Non-monotonic reasoning in operational research», écrit par Baxter (Stanford Univ.) et Randon (Bell Labs) est extrait (page 340) de l'ouvrage numéro 556473, intitulé «Advanced Topics in Decision Support Systems», publié par North-Holland en 1998.*

Suggestion. La phrase qui précède l'exemple devrait attirer votre attention.

Solution

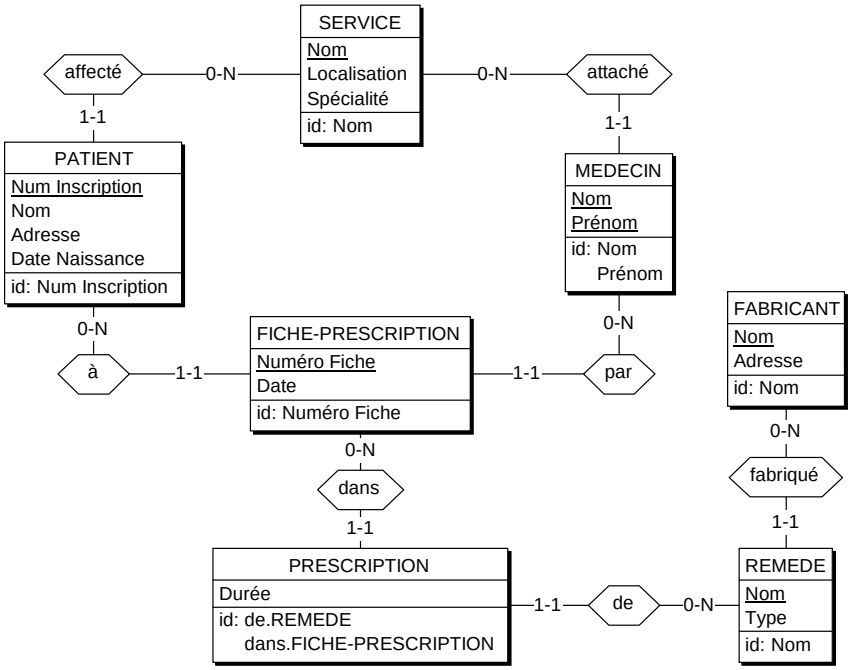


12.6 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

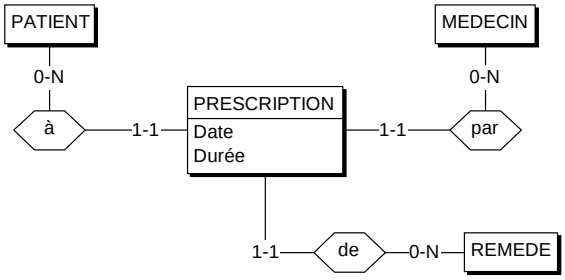
Les patients d'un hôpital sont répartis dans les services (caractérisés chacun par un nom identifiant, sa localisation, sa spécialité) de ce dernier. A chaque patient peuvent être prescrits des remèdes. Un remède est identifié par son nom et caractérisé par son type, son fabricant et l'adresse de ce dernier. Chaque prescription d'un remède à un patient est faite par un médecin à une date donnée pour une durée déterminée. On ne peut rédiger plus d'une prescription d'un remède déterminé pour un même patient le même jour. Chaque patient est identifié par un numéro d'inscription. On en connaît le nom, l'adresse et la date de naissance. Chaque médecin appartient à un service. Il est identifié par son nom et son prénom.

Suggestion. On sera particulièrement attentif à la notion de prescription.

Solution



Variante (mais problème d'identifiant)

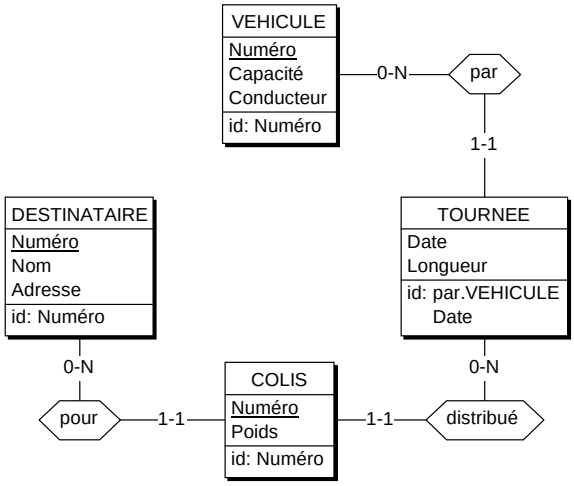


12.7 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

Une entreprise de distribution dispose d'un certain nombre de véhicules (identifiés par leur numéro et caractérisés par leur capacité et le nom du conducteur). Chaque jour, chaque véhicule effectue une (et une seule) tournée de distribution, d'une longueur déterminée. Durant cette tournée, le véhicule emporte des colis (décrits chacun par un numéro identifiant et un poids). Chaque colis doit être livré à un destinataire. Un destinataire est identifié par un numéro de destinataire et caractérisé par un nom et une adresse.

Suggestion. Attention à l'identifiant des tournées.

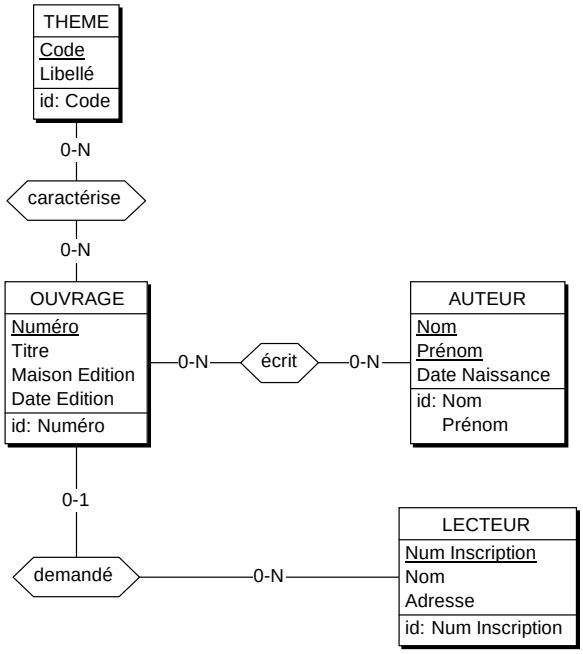
Solution



12.8 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

Une bibliothèque contient des ouvrages, repérés par un numéro unique et caractérisés chacun par son titre, sa maison d'édition et sa date d'édition. Un ouvrage peut avoir été écrit par un ou plusieurs auteurs. Un auteur est identifié par son nom et son prénom; il est caractérisé par sa date de naissance. Un ouvrage peut en outre être caractérisé par un ou plusieurs thèmes. Il existe un répertoire de thèmes standard, chaque thème étant décrit par un code concis (par exemple ROM.HIST) et son libellé en clair (par exemple ROMAN HISTORIQUE). Tous les codes concis sont différents. Lorsqu'on caractérise un ouvrage, on lui attribue le ou les thèmes qui le décrivent le mieux. Il se peut qu'un ouvrage ait été acquis suite à la demande personnelle d'un lecteur. Un lecteur est identifié par son numéro d'inscription et caractérisé par son nom et son adresse.

Solution

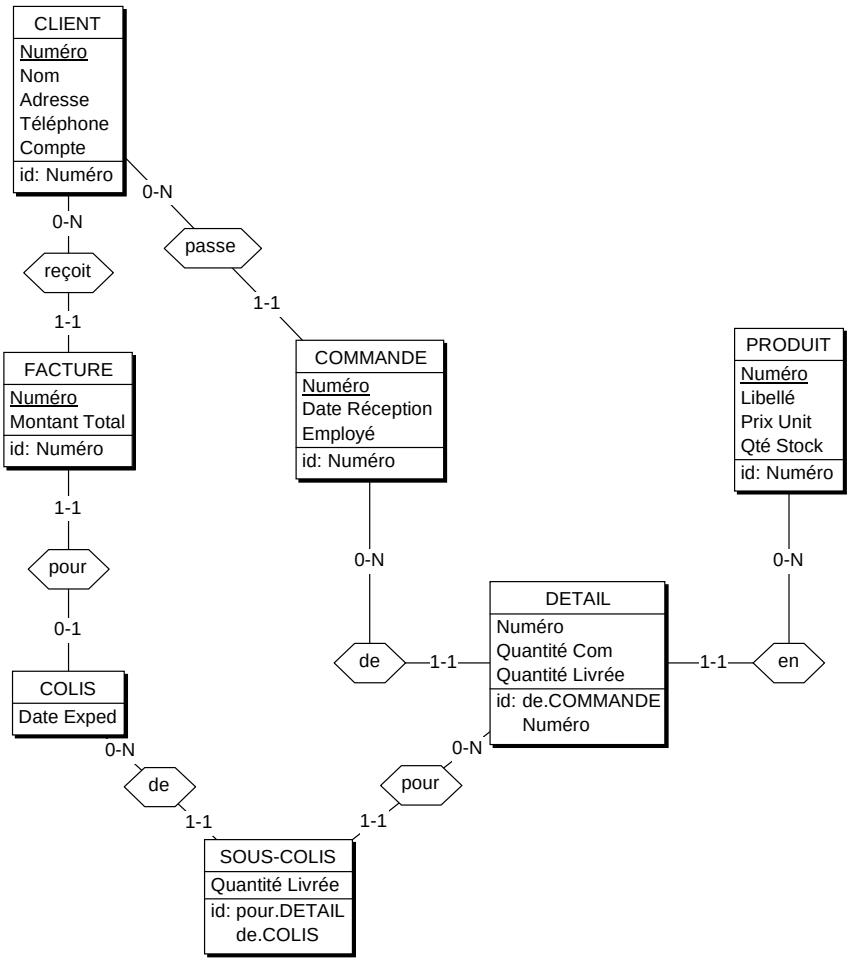


12.9 Proposer un schéma conceptuel qui représente le domaine d'application suivant :

Les clients de l'entreprise sont caractérisés par leur nom, leur adresse et le montant de leur compte. Ils reçoivent un numéro qui permet de les distinguer. Un client a passé un certain nombre de commandes, caractérisées chacune par un numéro qui les identifie, une date de réception et l'identification de l'employé qui l'a introduite. Chaque commande contient un certain nombre de détails de commande numérotés, qui spécifient chacun le produit commandé, la quantité désirée, la quantité déjà livrée et la date de la dernière livraison. Un produit est identifié par un numéro et caractérisé par un libellé et un prix unitaire. Les clients reçoivent des factures. Chaque facture correspond à un colis expédié à une date déterminée. Un colis comporte, pour un ou plusieurs détails de commande, la quantité expédiée (tout ou partie de la quantité commandée) en fonction de la disponibilité du produit. Une facture comporte un numéro identifiant, la date d'expédition du colis et un montant total. Elle spécifie aussi la composition du colis et les coordonnées du client : nom, adresse et téléphone. L'étiquette du colis reprend le numéro de sa facture, sauf lorsqu'il ne contient que des produits cadeaux. Les sous-colis d'un colis correspondent à des détails de commande distincts.

Suggestion. Attention aux propositions redondantes. Cet énoncé fait penser à un schéma bien connu : prudence tout de même.

Solution

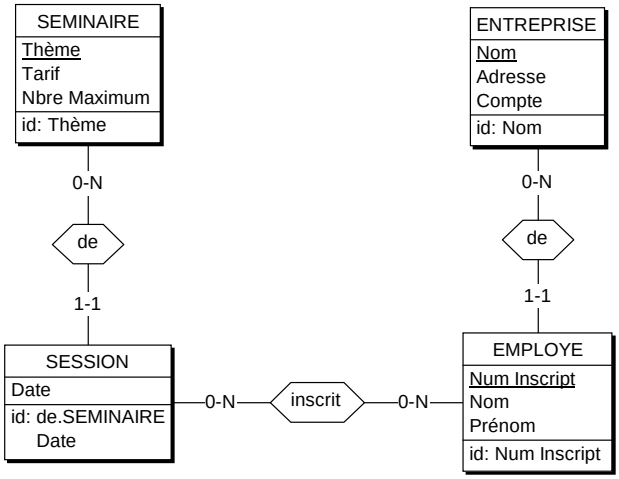


12.10 Construire un schéma conceptuel correspondant au fonctionnement d'une société de formation (version de base).

Une société de formation désire informatiser la gestion des inscriptions aux sessions qu'elle organise, ainsi que la facturation. Il existe un certain nombre de séminaires de formation, chacun consacré à un thème différent et facturé à un tarif déterminé. Un séminaire peut être organisé plus d'une fois, ce qui correspond à autant de sessions. Les sessions d'un séminaire se tiennent à des dates différentes. Des entreprises inscrivent certains de leurs employés à certaines sessions. Il existe un nombre maximum de participants pour les sessions de chaque séminaire (quelle que soit la date de la session). Tous les mois, la société facture à chaque entreprise concernée le montant correspondant à la participation de ses employés aux sessions du mois écoulé.

Suggestion. On ajoutera les attributs que le bon sens suggère pour permettre d'effectuer la facturation.

Solution



12.11 Construire un schéma conceptuel correspondant au fonctionnement d'une société de formation (version étendue).

Une société de formation désire informatiser la gestion des inscriptions aux sessions qu'elle organise, ainsi que la facturation. Il existe un certain nombre de séminaires de formation, chacun consacré à un thème différent et facturé à un tarif déterminé. Un séminaire peut être organisé plus d'une fois, ce qui correspond à autant de sessions. Les sessions d'un séminaire se tiennent à des dates différentes. Des entreprises inscrivent certains de leurs employés à certaines sessions. Lors d'une inscription, qui s'effectue à une certaine date, l'entreprise indique, pour chaque employé concerné, le nom, le prénom, le numéro de téléphone, le matricule interne à l'entreprise et la session à laquelle il est inscrit. Dès sa première inscription, l'employé devient un client de la société de formation et reçoit un numéro unique. Ultérieurement, l'inscription d'un employé à une session peut être annulée ou transférée vers un autre employé. Si l'annulation est effectuée moins d'une semaine avant la date de la session, un montant forfaitaire est cependant dû. Il existe un nombre maximum de participants pour les sessions de chaque séminaire (quelle que soit la date de la session). Chaque séminaire est pris en charge par un ou plusieurs instructeurs, mais ceux-ci ne sont pas nécessairement connus au moment où on prend les premières inscriptions, et peut utiliser un ou plusieurs ouvrages de référence. Tous les mois, la société facture à chaque entreprise concernée le montant correspondant à la participation de ses employés aux sessions du mois écoulé. On vérifiera que les informations répertoriées sont suffisantes pour rédiger la brochure commerciale de la société et pour produire et gérer les factures.

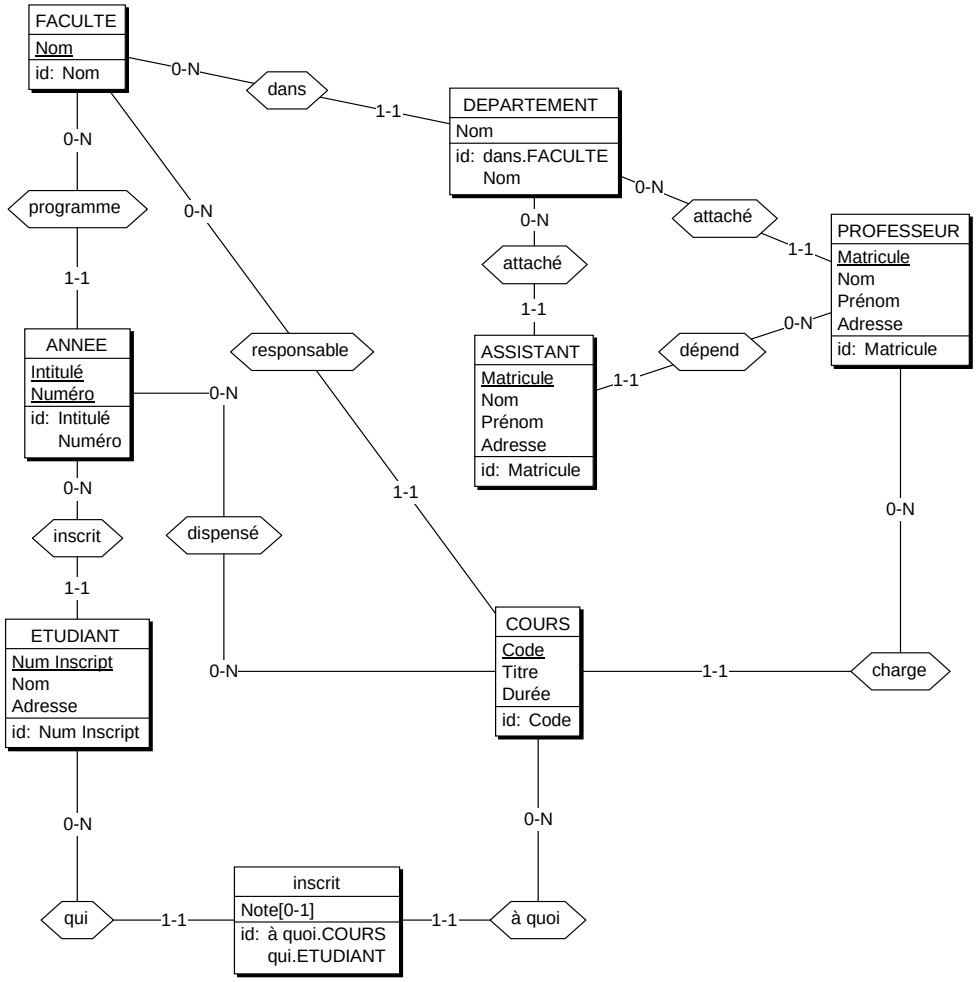
12.12 Construire un schéma conceptuel décrivant la structure d'une institution universitaire. On retiendra les faits suivants.

L'institution est constituée de facultés. A chaque faculté sont rattachés des professeurs et des assistants (qui dépendent chacun d'un professeur). Ces personnes sont regroupées en départements au sein de la faculté. Le programme d'enseignement d'une faculté est décomposé en cycles (premier ou Bachelor[2], deuxième ou Master, troisième ou doctorat) eux-mêmes comportant des années d'études (2e Bachelor en chimie, 1r Master en sciences politiques et Sociales, etc.) et est constitué de cours, dispensés dans une ou plusieurs années d'études, voire même dans d'autres facultés. Un cours, d'une durée déterminée, est pris en charge par un professeur. Chaque étudiant peut être inscrit à certains cours (on admet l'existence de dispenses, cours supplémentaires, etc.). Si l'étudiant a subi un examen relatif à un cours, on lui attribue la note qu'il a obtenue.

Remarque. Cet énoncé est (in)volontairement ambigu et incomplet. On explicitera les hypothèses choisies pour préciser l'énoncé.

2. Le choix de la nomenclature anglaise évite toute confusion avec les acceptions anciennes (mais toujours vivaces) des termes *baccalauréat* et *maîtrise.*

Solution

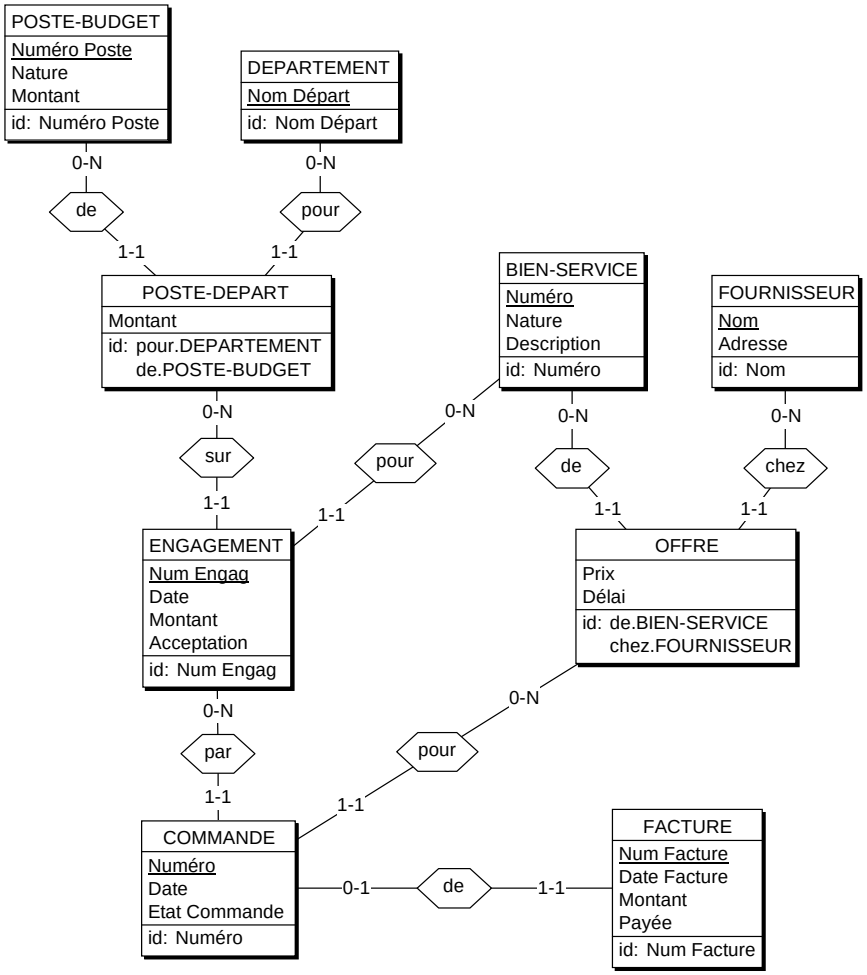


12.13 Construire un schéma conceptuel correspondant à une gestion budgétaire élémentaire décrite comme suit.

Une entreprise est constituée de différents départements. Le budget de l'entreprise est décomposé en postes de natures distinctes (petit matériel, déplacements, biens d'équipement, logiciels, frais de personnel, etc.). Chaque département reçoit une part propre de chacun de ces postes. Toute demande de dépense doit suivre la procédure suivante : le département désirant faire l'acquisition de biens ou consommer un service fait une demande d'engagement d'un certain montant, pour une nature de dépense relative au budget qui lui est propre; cette demande est vérifiée puis, si elle est acceptée, fait l'objet d'une commande auprès du fournisseur; à ce moment, le montant engagé est soustrait du poste budgétaire propre à ce département; à la réception du bien ou service, la facture est vérifiée puis

payée. On admet qu'une commande puisse être annulée. Le montant engagé est alors à nouveau disponible.

Solution

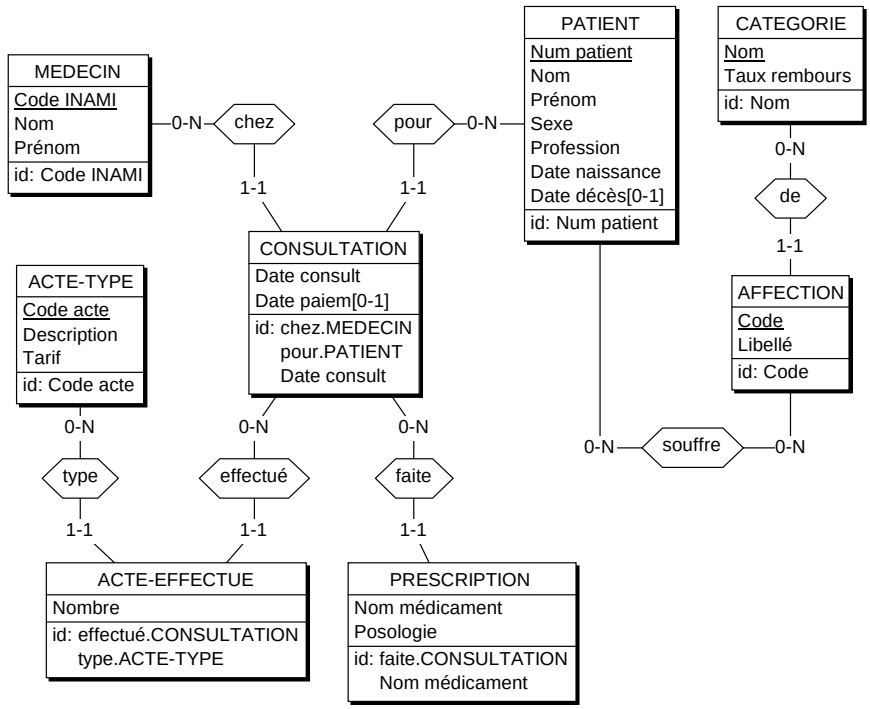


12.14 Construire le schéma conceptuel décrivant un cabinet de médecins.

Chaque médecin reçoit des patients en consultation. Lors de chaque consultation de chaque patient, un certain nombre d'actes médicaux sont effectués par le médecin. Chaque acte est répertorié dans une liste standard qui en donne le tarif. C'est ainsi que le médecin calcule le prix de la visite. Le patient peut payer lors de sa visite ou plus tard. Toute consultation non payée au bout d'un mois fait l'objet d'un rappel de paiement. Les sommes perçues durant chaque mois constituent le revenu du médecin. Cette information lui permet de remplir aisément sa déclaration fiscale annuelle.

Le médecin connaît la date de naissance, le sexe et la profession de chaque patient. Il sait également si le patient souffre d'un certain nombre d'affections réparties selon des classes standards. Lors de chaque visite d'un patient, le médecin inscrit la date, les prestations et les médicaments prescrits. Si un patient est décédé, il indiquera la date du décès.

Solution

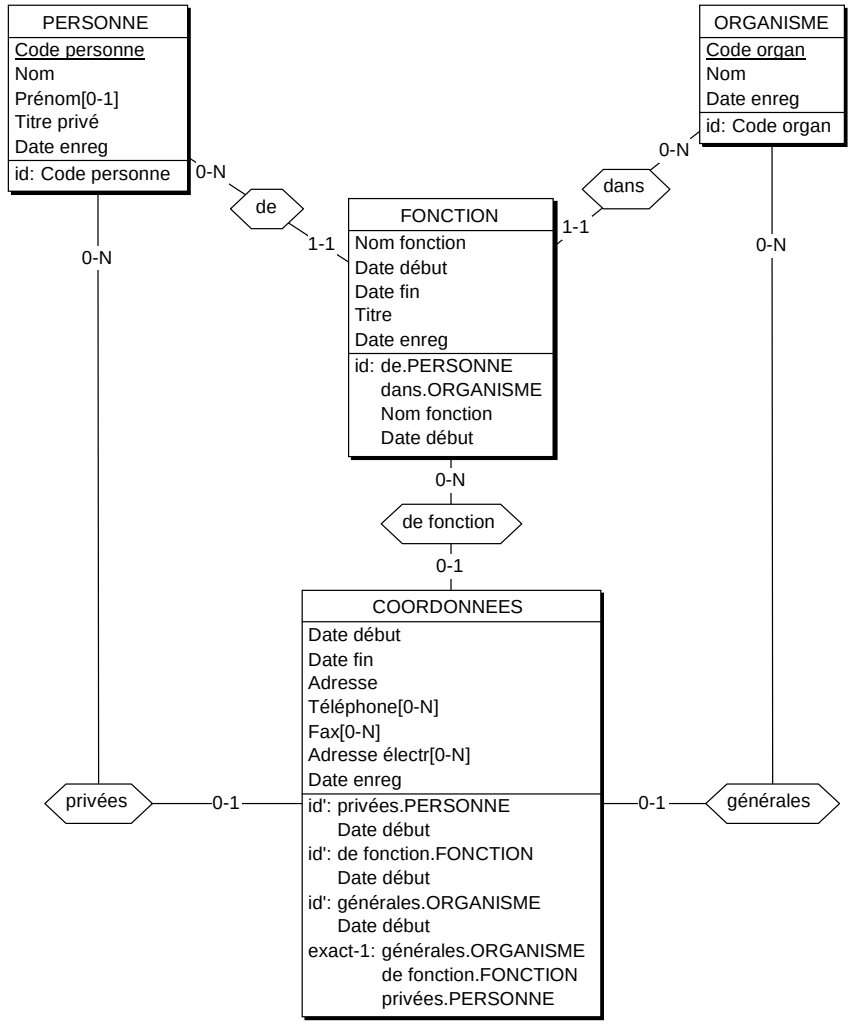


12.15 Un *carnet d'adresses* est une petite base de données qui ne semble pas poser de problèmes particuliers. Voire ! Proposer un schéma conceptuel des informations de contact décrites dans l'énoncé ci-dessous.

Pour toute personne, on reprend son nom, son prénom (lorsqu'il est connu), son titre (M., Mme, Maître, etc.), son adresse, son (ses) numéro(s) de téléphone fixe ou mobile et/ou de fax, son adresse électronique. Si la personne occupe une fonction dans une entreprise (ou organisme), on indique aussi le nom et l'adresse de celle-ci. Une personne peut avoir des coordonnées privées et des coordonnées professionnelles. Elle peut aussi occuper plus d'une fonction, dans la même entreprise ou dans des entreprises différentes, et pour chacune de ces fonctions, avoir des coordonnées différentes. Dans certains cas, ces coordonnées sont tout simplement celles de l'entreprise. Il est fréquent qu'une personne change d'adresse, de fonction ou téléphone. Il est important de pouvoir retrouver les caractéristiques d'une personne dans le passé (où travaillait Dupont l'année dernière, et quel est le

fax de son ancien employeur ?). On veut aussi savoir depuis quand chaque information a été enregistrée, pour en évaluer la validité.

Solution



12.16 Construire le schéma conceptuel décrivant des compétitions sportives.

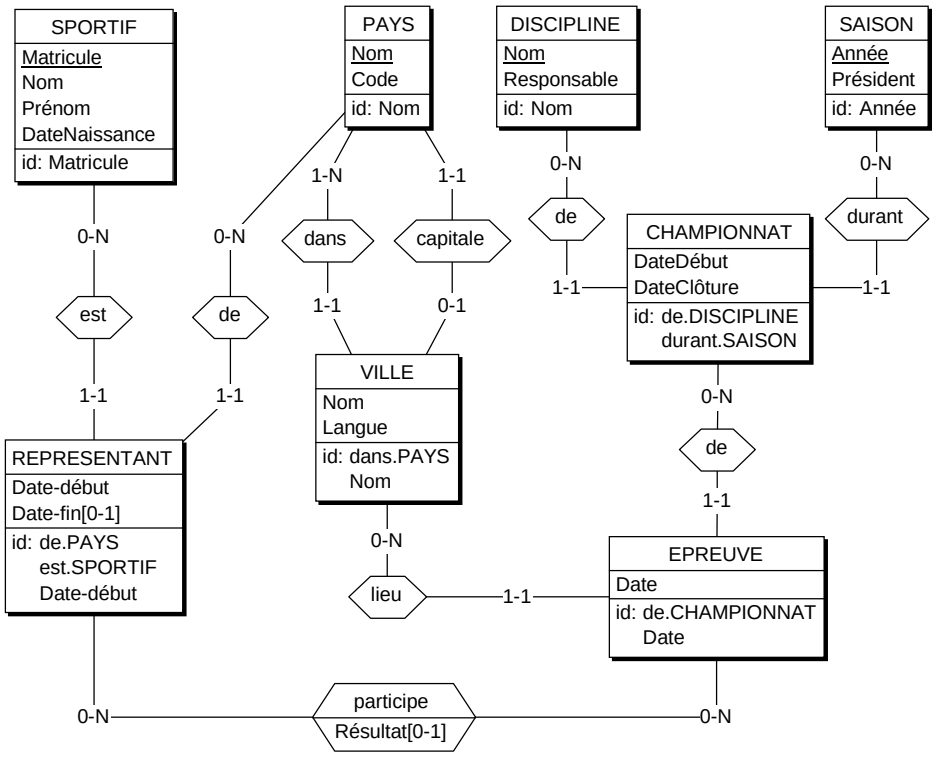
Chaque année est organisée une saison sportive, sous la présidence d'une personnalité de renom. Une saison est constituée d'une série de championnats, couvrant une certaine période, chacun consacré à une discipline sportive (nom identifiant et responsable). Il y a un seul championnat par saison et par discipline. Durant un championnat sont organisées, à des dates différentes pour un même championnat, des épreuves, localisées chacune dans une ville. Une ville porte un nom; on y parle une langue principale. Une ville est située dans un pays (nom, code). Les villes

d'un pays ont des noms distincts, mais rien n'interdit que deux pays aient des villes de même nom. Chaque pays a une capitale, qui est une ville de ce pays.

Des sportifs (matricule, nom, prénom, date de naissance) représentent des pays. Durant une période déterminée, un sportif représente un seul pays. Il peut alors participer à une ou plusieurs épreuves, au terme de chacune desquelles il obtient un résultat.

On vérifiera que le schéma permet de répondre à des questions telles que les suivantes : Qui a obtenu la médaille de bronze en 110 m haies en 2001 ? Quel pays n'a obtenu aucune médaille en 1999 ? Quels sont les sportifs qui ont remporté une médaille d'or dans la capitale du pays qu'ils représentaient ?

Solution



12.17 Normaliser le schéma de la figure 1.3. Parmi les attributs de DEPARTEMENT, on observe les dépendances suivantes :

Entreprise ⟶ Adresse, Responsable;

Responsable ⟶ Téléphone

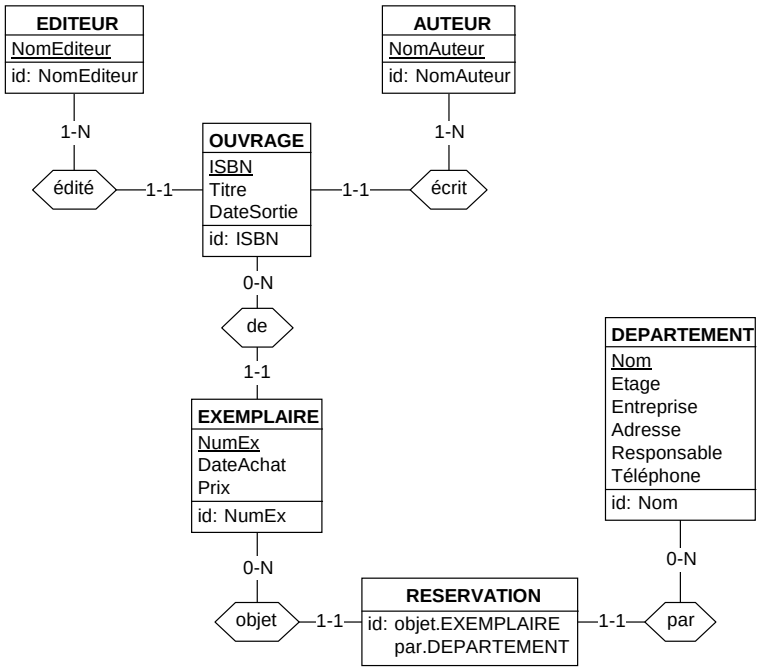
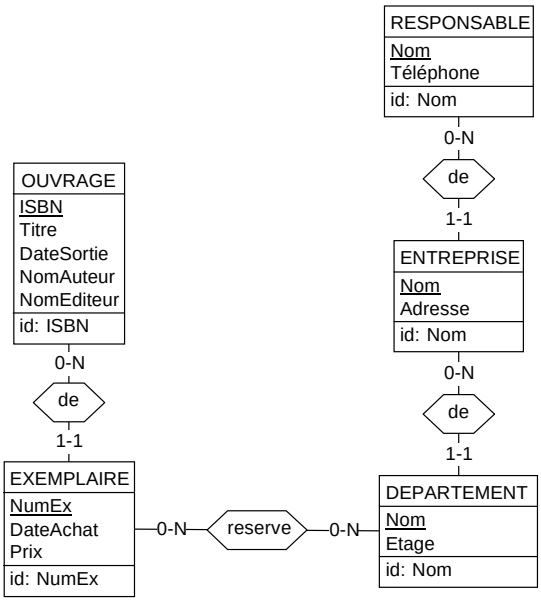


Figure 1.3 - Un schéma à normaliser

Solution



12.18 Montrer que les deux schémas de la figure 1.4 sont équivalents. On adoptera d'abord une approche intuitive, basée sur un graphe de populations représentatif (figure 13.15). On se servira ensuite des transformations présentées aux figures 13.6 et 13.8.

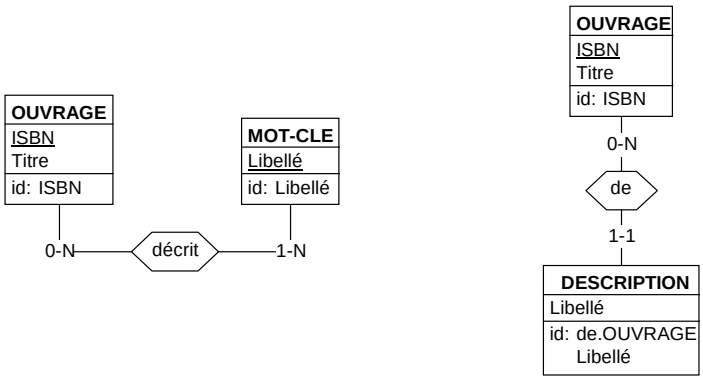
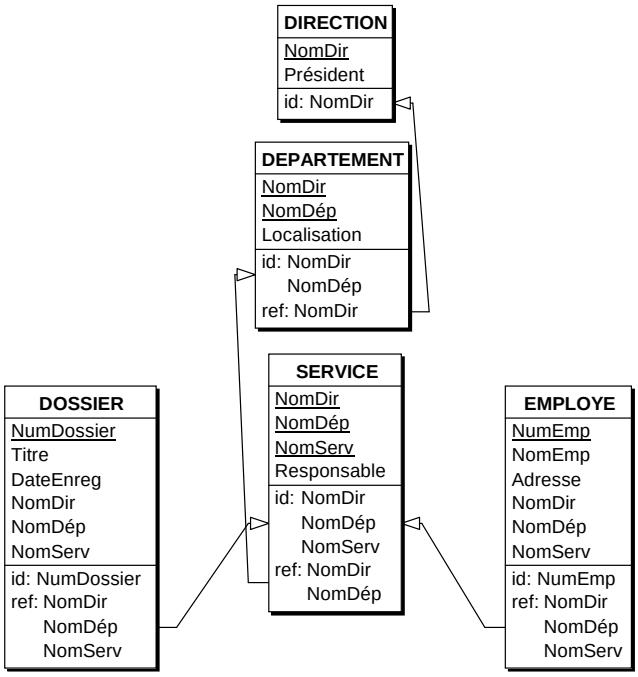


Figure 1.4 - Equivalence de schémas

A.13 CHAPITRE 13 - PRODUCTION DU SCHÉMA DE LA BASE DE DONNÉES

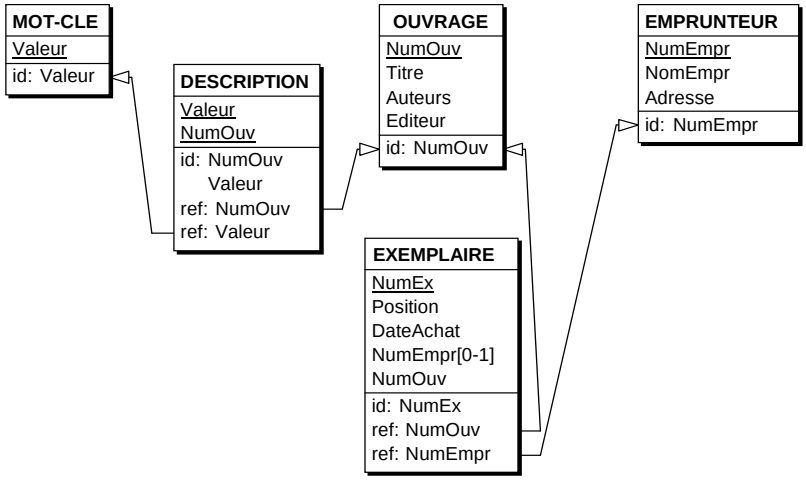
13.1 Construire le schéma des tables correspondant au schéma conceptuel d'une structure administrative de la figure 11.25.

Solution



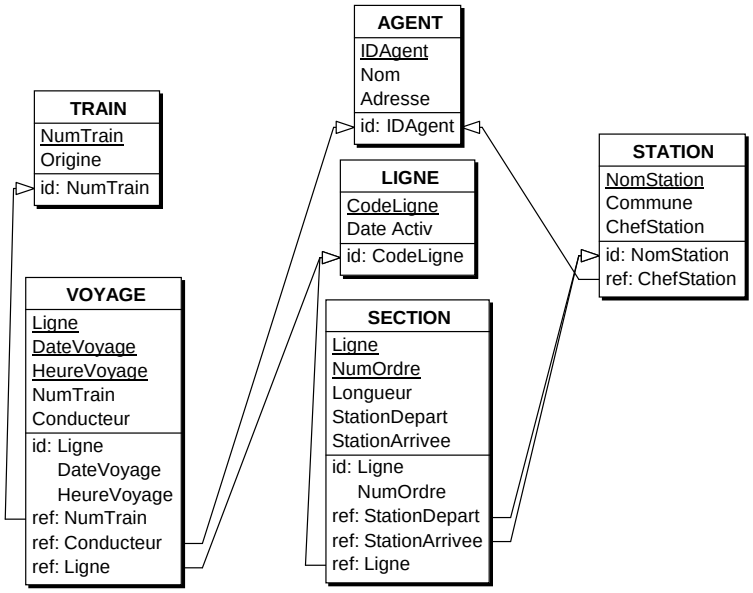
13.2 Construire le schéma des tables correspondant au schéma conceptuel d'une gestion de bibliothèque de la figure 11.26.

Solution



13.3 Construire le schéma des tables correspondant au schéma conceptuel représentant des voyages en train de la figure 11.27.

Solution



13.4 Dériver un schéma de tables à partir du schéma conceptuel de la figure 1.5.[3]

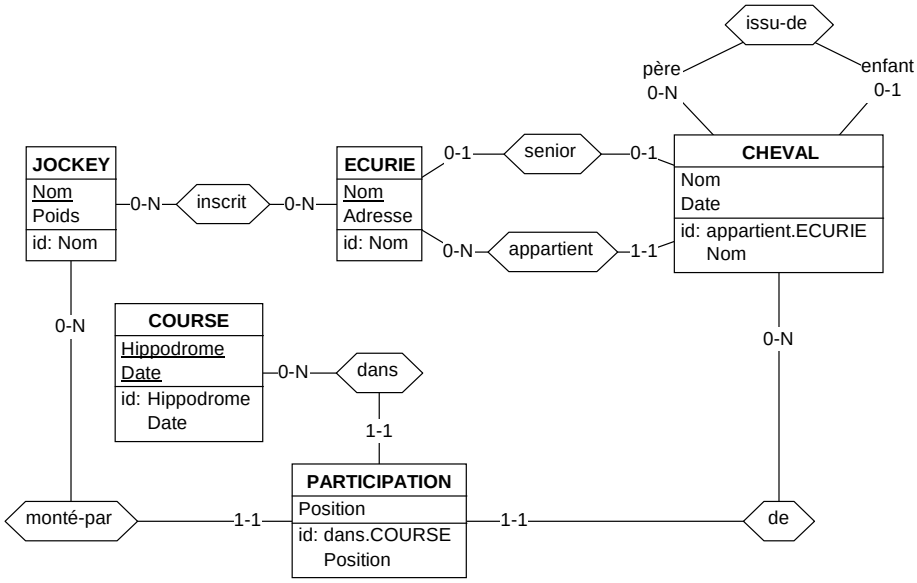
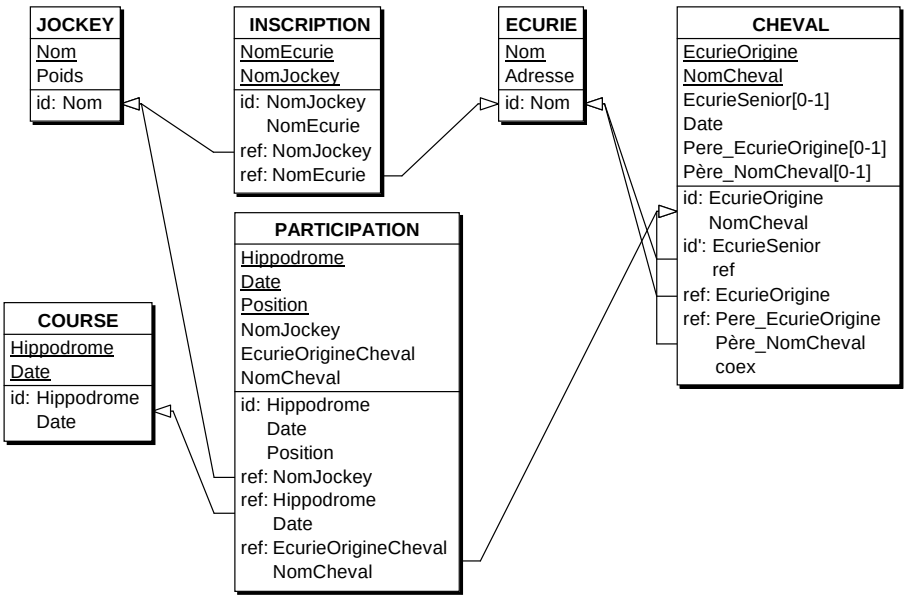


Figure 1.5 - Un schéma conceptuel à traduire

Solution



3. On notera que le schéma ne dit rien sur les liens éventuels entre les types d'associations senior et appartient. Il conviendrait que la traduction demandée ne fasse pas non plus d'hypothèse sur ces liens.

13.5 Traduire en un schéma de tables le schéma conceptuel de l'exercice 13.5. Garnir les tables avec les données qui correspondent à l'exemple décrit.

13.6 Traduire en structure de tables le schéma conceptuel de l'exercice 13.13. Ecrire les requêtes qui aideront le médecin dans ses tâches d'analyse de données. Ses questions les plus fréquentes sont les suivantes :
- le nombre de cas dans chacune des classes d'affections;
- la corrélation entre la profession et chacune de ces classes d'affections;
- la corrélation entre la localité de résidence et chacune de ces classes d'affections;
- la corrélation entre l'âge et chacune de ces classes d'affections;
- la corrélation entre la localité de résidence et l'âge des patients;
- la répartition des âges de décès selon la classe d'affection;
- la répartition des âges de décès selon la profession.

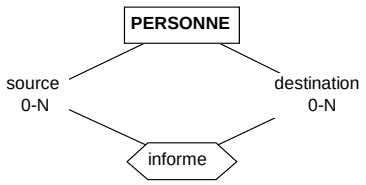
13.7 Justifier les modes `on delete` et `on update` du code SQL de la section 14.9.

A.14 CHAPITRE 14 - MÉTHODOLOGIE DES BASES DE DONNÉES

Néant

A.15 CHAPITRE 15 - LE MODÈLE ENTITÉ-ASSOCIATION ÉTENDU

15.1 On considère le type d'associations cyclique informe ci-dessous, qui représente les flux d'information entre des personnes. Si (p1, p2) ∈ PERSONNE, alors la personne p2 reçoit de l'information de la personne p1. On précise que ce type d'associations correspond à une relation transitive : si (p1, p2) et (p2, p3) existent, alors (p1, p3) existe également. Discutez de l'utilité de représenter par les instances toutes les associations ou seulement celles de la réduction transitive (qui ne sont pas obtenues par transitivité).

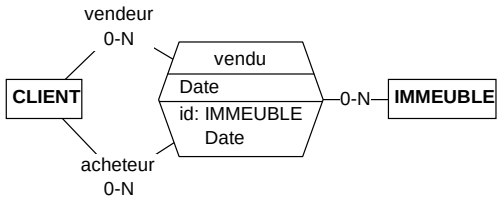


15.2 Construire un tableau similaire à celui de la figure 16.32 pour les types d'associations binaires.

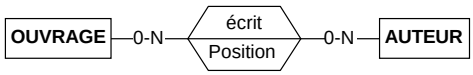
15.3 Le type d'associations cyclique vendu de la figure 16.8 n'est pas satisfaisant, car il impose certaines contraintes aux possibilités de vente. On suggère d'ajouter un attribut Date Vente. Compléter le schéma en introduisant cet attribut et en définissant l'identifiant qui semble le plus approprié.

Solution

Ce schéma interdit à un vendeur de vendre plus d'une fois un même immeuble à un même acheteur. Situation rare mais possible ! En introduisant la *date de vente*, et en admettant qu'un immeuble ne puisse être vendu deux fois le même jour, il vient :

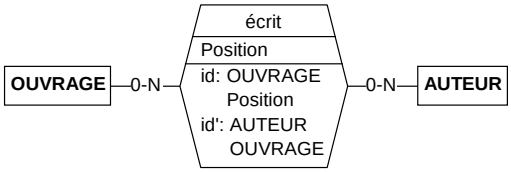


15.4 On considère le schéma ci-dessous, exprimant qu'un auteur apparaît dans une certaine position dans la liste des auteurs d'un ouvrage. Compléter ce schéma de manière à rendre explicite les deux règles métier : *un auteur n'apparaît qu'une seule fois pour un ouvrage*, et *il n'y a qu'un seul auteur à une position déterminée pour un ouvrage*. Ce schéma est-il normalisé ?



Solution

Cette question a été abordée dans l'exercice 3.5, où on a conclu que le schéma était normalisé.



15.5 Représenter par un schéma un ensemble de personnes qui peuvent être unies par les liens du mariage. On représentera les faits suivants :

- il n'est pas interdit à une personne de se marier plusieurs fois,
- à une date donnée, une personne ne peut avoir qu'un seul conjoint,
- il n'est pas obligatoire d'être marié à tout instant,
- il n'est pas interdit à une personne d'épouser une même personne plus d'une fois.

15.6 On considère le schéma de la figure 16.52. Combien ce schéma comporte-t-il de cycles ? Comment un type d'associations n-aire est-il pris en compte dans le repérage des cycles ? L'un de ces cycles fait-il l'objet de contraintes d'intégrité cycliques ?

15.7 On considère les deux types d'associations r entre A et B et s entre B et C. On considère aussi le type d'associations rs défini comme la composition de r et s (figure 1.6) Calculer les cardinalités des rôles de rs à partir de celles des rôles de r et de s. On procédera en deux étapes :

1. on déterminera la classe fonctionnelle (1:N, 1:1, N:1 ou N:N) à partir de celles de r et de s,
2. on calculera les cardinalités [v-w] et [x-y] à partir de celles de r et de s.

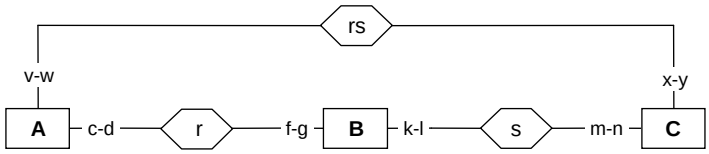
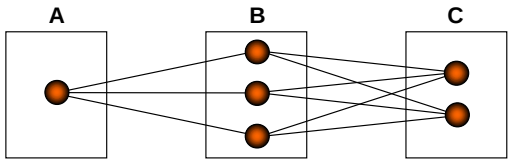


Figure 1.6 - Etude de la composition de deux types d'associations

Avant de crier prématurément victoire, on vérifiera que les formules obtenues, appliquées aux valeurs c = 3 et k = 2, décrivent correctement le schéma d'occurrences ci-dessous :



Solution (corrigée le 4/1/2010)

On ne traite ici que la question 2. La question 1 est reportée au chapitre 11.

Première approche

Une entité A quelconque apparaît dans au plus **d** associations r et est donc associée à **d** entités B au plus. Chacune de celles-ci apparaît dans au plus **l** associations s et donc est associée à **l** entités C au plus. On suggère donc **w** = **d** x **l**, et, par symétrie, **y** = **m** x **f**.

Pour les cardinalités minimales **v** et **x**, on pourrait a priori appliquer un raisonnement similaire : une entité A apparaît dans au moins **c** associations r et est donc associée à au moins **c** entités B. A chacune de celles-ci correspondent au moins **k** entités de C. On aurait donc **v** = **c** x **k**, et, par symétrie, **x** = **n** x **g**.

Malheureusement, ce dernier raisonnement est trop simpliste, comme l'illustre le schéma d'instances de l'énoncé, qui représente des instances valides du schéma pour **c** = **3** et **k** = **2**. On y observe qu'il existe bien **c** x **k** = **6** associations s, mais qu'une entité A n'est associée via rs qu'à **2** entités C, et non **6** au moins comme le voudrait la règle ci-dessus.

Analyse

Il apparaît en effet que deux entités B distinctes peuvent être associées à une même entité C. Dans un tel cas, le nombre d'entités C associées à une entité A pourra être inférieur à **c** x **k**. La valeur **c** x **k** représente donc la valeur maximale de **v**, mais certaines configurations admettent des valeurs inférieures. Il n'est pas difficile de déterminer ce qui caractérise ces situations :

pour qu'une même entité C puisse être partagée par plusieurs entités B, il faut que celle-ci puisse apparaître dans plusieurs associations s, et donc il faut que **m** > **1**.

La règle **v** = **c** x **k** n'est donc correcte que lorsque **c = 0** ou **k = 0** (auxquels cas **v = 0**), ou lorsque **m** = **1**. Elle doit être affinée lorsque **c** et **k > 0** et **m** > **1**, ce que nous allons faire.

Etudions d'abord jusqu'où nous pouvons légalement réduire le nombre d'entités C. Nous savons qu'à toute entité A correspondent (au moins) **c** x **k** associations s. Or, une entité C ne peut apparaître dans plus de **m** associations s. Donc ces **c** x **k** associations s exigent au moins $\lceil$**c** x **k/m**$\rceil$ entités C distinctes. On a donc **v** $\geq \lceil$**c** x **k/m**$\rceil$.

Observons ensuite que cette réduction est limitée par une autre contrainte : puisqu'une entité B doit apparaître dans au moins **k** associations s, il faut pour construire celles-ci au moins **k** entités C distinctes, le nombre total d'entités C ne peut être inférieur à **k**. On a donc aussi **v** $\geq$ **k**.

On en conclut que **v** $\geq$ max($\lceil$**c** x **k/m**$\rceil$**, k)** lorsque **c > 0** et **k > 0** et **m** > **1**.

Nous disposons à présent de deux valeurs pour **v** : une valeur maximale (**c** x **k**), valable lorsque **c** = 0 ou **k** = 0 ou **m** =1 et une valeur minimale (max($\lceil$**c** x **k/m**$\rceil$, **k**)) valable dans les autres cas. On unifie ces deux cas par la règle

v = min(**c** x **k**, max($\lceil$**c** x **k/m**$\rceil$, **k**)).

Conclusion

On peut donc écrire :

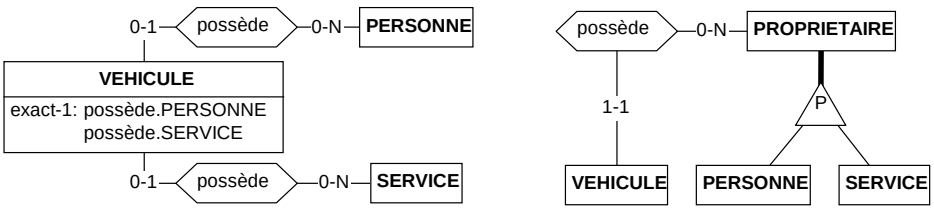
v = min(**c** x **k**, max($\lceil$**c** x **k/m**$\rceil$, **k**))
w = **d** x **l**
x = min(**n** x **g**, max($\lceil$**n** x **g/d**$\rceil$, **g**))
y = **m** x **f**

Remarque

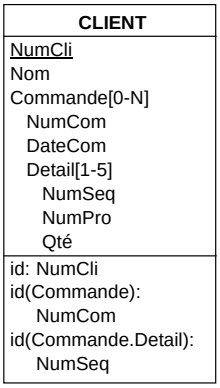
Afin que les tailles des populations des types d'entités n'imposent pas de contraintes sur les cardinalités, on suppose que ces tailles sont très grandes vis-à-vis de d, g, l, n.

15.8 Proposer deux formes équivalentes du type d'associations multi-type d'entités de la figure 16.9.

Solution

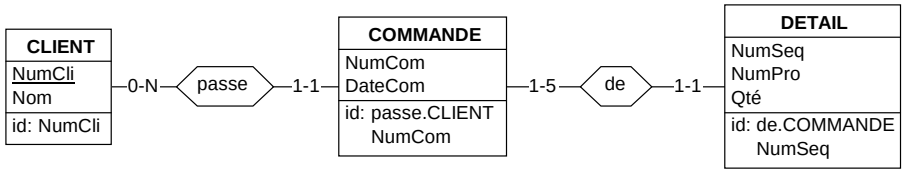


15.9 Un attribut de structure trop complexe peut être l'indice d'une mauvaise modélisation. Montrer que l'exemple ci-contre en est une illustration. Proposer une variante équivalente plus appropriée.



Solution

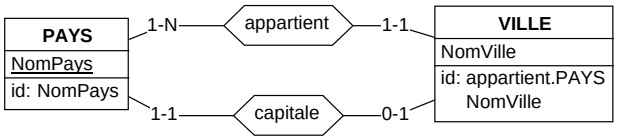
L'attribut complexe Commande est extrait sous la forme du type d'entités COMMANDE. Son attribut complexe Détail de ce dernier est traité de la même manière. On obtient dès lors le schéma ci-dessous :



15.10 On considère le schéma de droite de la figure 16.40, représentant des nomenclatures de produits. On a convenu que le graphe des instances ne comportait pas de circuits (16.12.4). Cette propriété garantit par exemple que des procédures récursives travaillant sur les instances ne boucleront pas.

Admettons maintenant que certains produits finis sont réinjectés dans le processus de fabrication, comme c'est le cas dans la métallurgie et dans l'industrie verrière, où les chutes et les pièces non conformes sont recyclées, de sorte qu'un produit fini est aussi une matière première. Modifiez le schéma de manière (1) à tenir compte de l'existence de tels circuits et (2) à conserver la propriété d'acyclicité initiale.

15.11 Compléter le schéma ci-dessous des contraintes qui paraissent évidentes.



Solution

Si on admet que la capitale d'un pays appartient à ce dernier, alors il faut imposer la contrainte capitale $\subseteq$ appartient.

15.12 Compléter le schéma de la figure 1.7 en calculant les grandeurs inconnues, notées "?".

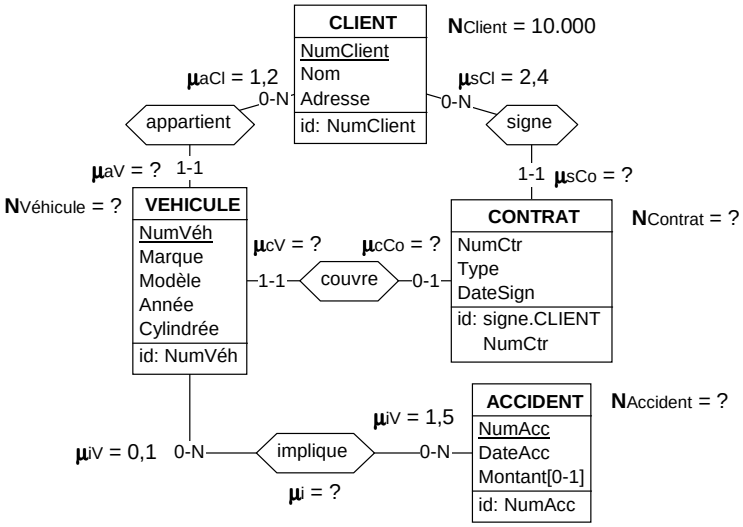


Figure 1.7 - Calcul des tailles des populations

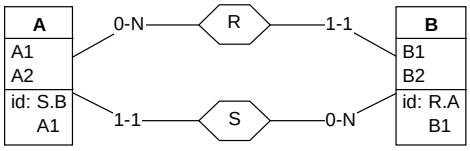
Solution

En exploitant les relations de la section 15.8.7, il vient :

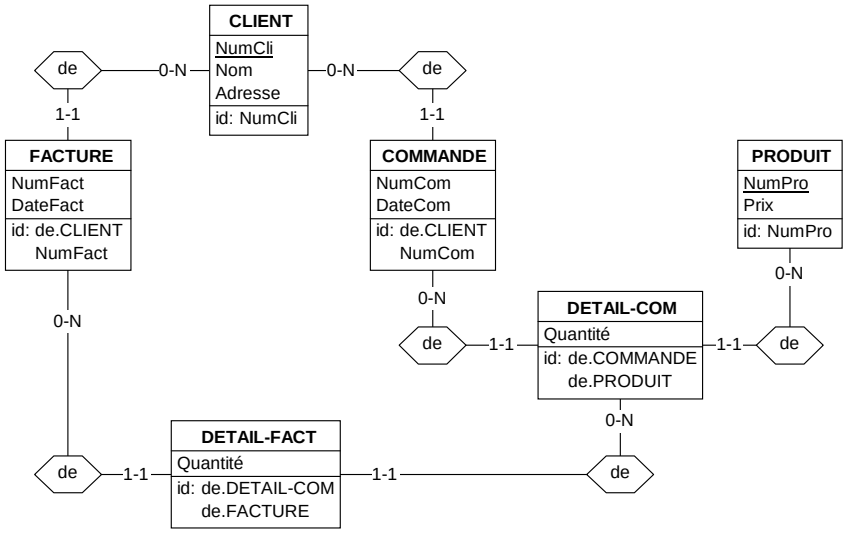
appartient :	$\mu_{aV} = 1$	ACCIDENT :	$\mathbf{N}_{Accident} = 800$
signe :	$\mu_{sCo} = 1$	couvre :	$\mu_{cV} = 1$
VEHICULE :	$\mathbf{N}_{Véhicule} = 12.000$	couvre :	$\mu_{cCo} = 0,5$
CONTRAT :	$\mathbf{N}_{Contrat} = 24.000$	implique :	$\mu_{i} = 1.200$

15.13 Le schéma de la question 15.15 est-il le siège d'une contrainte cyclique ? Si oui, en écrire l'expression.

15.14 Analyser le schéma ci-dessous.



15.15 Indiquer pour chaque type d'entités du schéma ci-dessous les attributs pour lesquels on doit fournir une valeur de manière à identifier une entité de ce type. Par exemple, pour identifier une entité CLIENT, il faut fournir une valeur de CLIENT.NumCli, tandis que pour une entité FACTURE, il faut fournir une valeur de de.CLIENT.NumCli et une valeur de FACTURE.NumFact[4].



Solution

Pour identifier ...	**il faut connaître les valeurs de ...**
CLIENT	CLIENT.NumCli
FACTURE	de.CLIENT.NumCli, FACTURE.NumFact
COMMANDE	de.CLIENT.NumCli, COMMANDE.NumCom
PRODUIT	PRODUIT.NumPro
DETAIL-COM	de.COMMANDE.de.CLIENT.NumCli, de.COMMANDE.NumCom, de.PRODUIT.NumPro
DETAIL-FACT	de.FACTURE.de.CLIENT.NumCli, de.FACTURE.NumFact, de.DETAIL-COM.de.COMMANDE.de.CLIENT.NumCli, de.DETAIL-COM.de.COMMANDE.NumCom, de.DETAIL-COM.de.PRODUIT.NumPro

L'identification de DETAIL-COM nécessite la connaissance de deux valeurs de NumCli.

4. Attention, ceci ne veut pas dire que le groupe {NumCli, NumFact} constitue un identifiant de FACTURE.

15.16 Proposer un méta-schéma pour les modèles SQL2 et SQL3.

15.17 Proposer un méta-schéma pour le modèle relationnel (chapitre 3).

15.18 Proposer un méta-schéma pour le modèle Entité-association étendu.

15.19 Imaginer un schéma équivalent à celui de la figure 16.40 (droite) mais utilisant un type d'associations de composition.

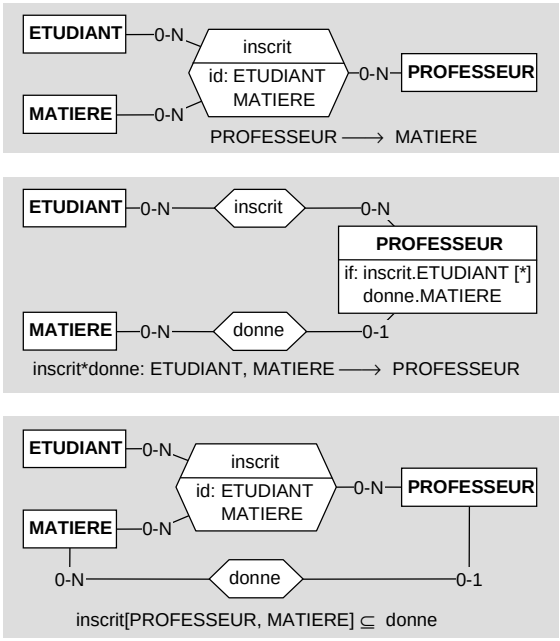
15.20 Le schéma de droite de la figure 16.48 décrit partiellement la composition d'un véhicule. Son interprétation est cependant loin d'être évidente. Le lecteur est invité à examiner les questions suivantes, et à modifier ou compléter le schéma si nécessaire.

- Que représente une entité MOTEUR ? Un type de moteur ou un moteur bien précis ?
- Une entité MOTEUR existe-elle indépendamment de l'entité VEHICULE dans la composition de laquelle elle entre ? Un moteur peut-il changer de véhicule ? Un véhicule peut-il changer de moteur ?
- Le type d'entités ROUE représente-t-il les *types de roue* disponibles ou bien toutes *roues* fabriquées ? Comment modifier le schéma pour tenir compte de l'autre interprétation ?

15.21 On a montré dans la section 16.18.9, point (c), que le type d'associations inscrit de la figure 16.38 n'était pas normalisé et qu'il pouvait se transformer en trois schémas équivalents. Dessiner ces schémas dans le modèle Entité-association en indiquant soigneusement toutes les contraintes d'intégrité.

Solution

En s'inspirant des solutions obtenues à la fin de la section 3.8.5, on obtient :



15.22 Définir la sémantique relationnelle du schéma 11.27 (*Voyages en train*).

15.23 Définir la sémantique relationnelle du schéma 16.17.

15.24 Définir la sémantique relationnelle du schéma 16.53.

15.25 On considère un type d'entités FEMME doté des attributs Nom et Prénom. Compléter ce schéma en y incluant la représentation des deux propriétés suivantes :

a) une femme peut être la mère d'autres femmes,

b) une mère donne à ses filles des prénoms distincts.

Solution

Il s'agit d'un exemple d'identifiant cyclique (voir section 15.10.2).

15.26 Représenter la structure et la croissance de certains arbres fruitiers (en l'occurrence des pommiers) qui obéissent aux lois suivantes. D'une manière générale, et en simplifiant à outrance, un arbre est formé d'axes (branches) dotés chacun d'une séquence de noeuds. Chaque axe (sauf l'axe principal, qui est le tronc) pousse au départ d'un noeud d'un axe plus ancien. La structure qui résulte de ces règles est, sans surprise, celle d'un arbre. On peut désigner univoquement un axe en précisant son axe père et le numéro du noeud dont il est issu sur cet axe[5].

Solution

Il s'agit d'un exemple d'identifiant cyclique (voir section 15.10.2).

15.27 Le méta-schéma de la figure 16.53 représente des identifiants qui traduisent l'unicité des noms des objets d'un schéma. Cependant, la contrainte qui veut que les *noms des rôles d'un type d'associations soient distincts* n'est pas traduite correctement si on inclut dans ces noms les noms par défaut. Corriger ce schéma pour tenir compte de ces derniers.

15.28 Considérons le tableau des contraintes d'existence de la section 16.11.3. Le lecteur quelque peu familier avec la logique aura reconnu dans ce tableau une table de vérité définissant complètement cinq fonctions logiques à deux variables (A et B). Sachant qu'il existe 16 fonctions logiques à deux variables, on observe que ce tableau n'est pas complet. On peut représenter chaque fonction par les valeurs présentes dans sa colonne lues de haut en bas. Ainsi, coex = [1001] et excl = [1110]. Etudier les autres fonctions : sont-elles utiles ? Existe-t-il des fonctions primitives, à partir desquelles il est possible de construire les autres ?

Solution

Le tableau ci-dessous complète celui de la section 16.11.3 en représentant les 16 fonctions. On attribue, en entête de chaque colonne, un numéro de 0 à 15 à chaque fonction. Ainsi, coex = [1001] reçoit le numéro 9 et excl = [1110] le numéro 14.

A	B	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	**0**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	**1**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	**0**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	**1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		-	-	-		-				-							-

On peut faire quelques observations intéressantes.

- Certaines fonctions n'ont aucune utilité. Par exemple, la fonction 0 ([0000]) rend le parent *non satisfiable* et la fonction 15 ([1111]) n'impose aucune contrainte. Nous pouvons les ignorer.
- De même, les quatre fonctions ne comportant qu'un seul signe 1 sont sans intérêt, puisqu'on peut toujours leur substituer une modification du schéma : supprimer A et/ou B ou rendre A et/ou B obligatoire. On ignore donc aussi les fonctions 1, 2, 4 et 8.

5. Problème aimablement communiqué par J-J. Claustriaux, professeur à la Faculté Agronomique de Gembloux.

- Restent ainsi les dix fonctions utiles 3, 5, 6, 7, 9, 10, 11, 12, 13 et 14. Certaines comporte trois signes 1 (7, 11, 13, 14) et les autres deux signes 1 (3, 5, 6, 9, 10, 12).
- On observe que toutes les fonctions à deux signes 1 peuvent être obtenues par une conjonction de deux fonctions à trois signes 1. Par exemple, [1010] = [1011] $\wedge$ [1110].
- On en conclut que les quatres fonctions (7, 11, 13, 14) sont primitives et permettent de reconstruire toutes les fonctions utiles. Elles correspondent aux contraintes "at-lst-1: A, B", "if: B, A", "if: A, B" et "excl: A, B".
- Notre modèle Entité-association comporte donc trois contraintes d'existence primitives **at-lst-1**, **if** et **excl**.

15.29 Etablir les tables de vérité des contraintes d'existence portant respectivement sur 3 et 4 composants facultatifs.

Solution

Cette question sera résolue pour les expressions à 2 et 3 composants à la section 22.5. On y montre qu'il existe une loi de formation qui permet d'extrapoler l'expression des contraintes de plus de 3 composants.

15.30 On considère les contraintes excl: A, B, C et coex: A, B, C. Exprimer ces contraintes en utilisant des contraintes à deux composants.

15.31 En examinant la table de vérité évoquée à l'exercice 15.28 et en considérant la fonction logique existe(x.E), qui renvoie vrai si le composant E de l'objet x existe et faux sinon, établir la contrainte, ou la conjonction de contraintes, équivalente à chacune des expressions suivantes :

équivalence : existe(x.A) = existe(x.B)

disjonction exclusive : existe(x.A) $\oplus$ existe(x.B)

implication : existe(x.A) $\Rightarrow$ existe(x.B)

disjonction : existe(x.A) $\vee$ existe(x.B)

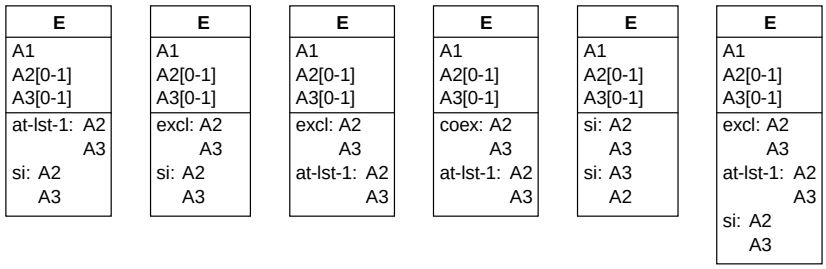
conjonction : existe(x.A) $\wedge$ existe(x.B)

15.32 Comment exprimer à l'aide des contraintes d'existence la propriété qui dit que *si A existe, alors B doit être absent*.

Solution

Si A est absent, la propriété est vraie. Si A est présent et B absent la propriété est vraie. Si A est présent et B l'est aussi la propriété est fausse. Cette propriété correspond donc à la fonction [1110], qui est l'expression de **excl: A, B**.

15.33 En utilisant les résultats de la question 15.28, simplifier chacun des schémas suivants.



A.16 CHAPITRE 16 - LES DIAGRAMMES DE CLASSES UML

16.1 Exprimer en OCL les contraintes manquantes dans les schémas 17.9 et 17.10.

16.2 Proposer un méta-schéma des diagrammes de classes de DB-UML.

16.3 Proposer un méta-schéma des diagrammes de classes d'UML.

A.17 CHAPITRE 17 - ANALYSE CONCEPTUELLE DU DOMAINE D'APPLICATION

17.1 *Toute personne possède deux parents biologiques, lesquels sont des personnes*. Discuter cette proposition dans le contexte de l'analyse conceptuelle.

Solution

Introduire dans le schéma, comme une analyse naïve pourrait le suggérer, un rôle enfant de cardinalité [2-2] rendrait le type d'entités PERSONNE **non satisfiable** (section 17.4.2/c). On se souviendra qu'un schéma conceptuel ne représente pas à proprement parler un domaine d'application, mais plutôt la connaisssance qu'on en a. Comme on ne connaît pas les parents biogiques de chaque personne, on dégradera la contrainte en [0-2].

17.2 Proposer un schéma conceptuel explicitant les relations existant parmi les *personnes* formant des *familles* et/ou des *ménages*. On veillera à représenter l'évolution de ces structures.

17.3 Construire le schéma conceptuel décrivant un réseau de distribution d'eau.

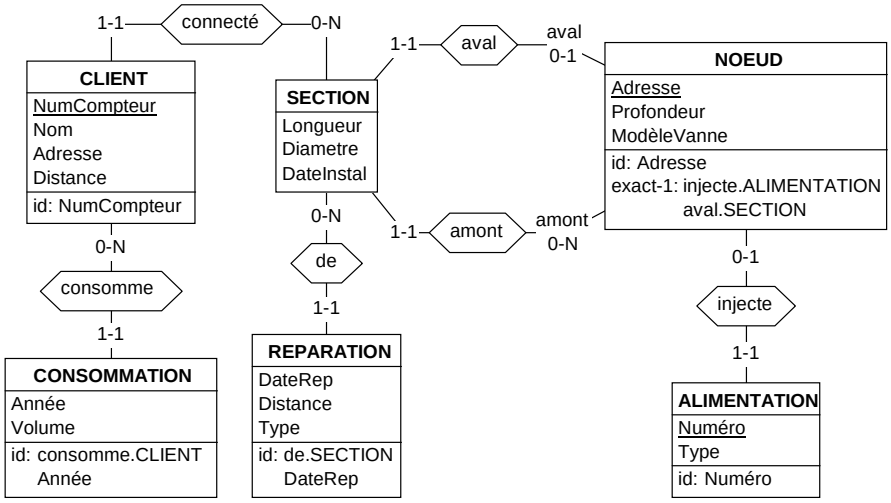
Un réseau de distribution d'eau est constitué de sections de canalisation. Une section relie deux nœuds : le nœud amont (alimentant) et le nœud aval (alimenté); l'eau s'écoule donc dans un sens déterminé, de l'amont vers l'aval. Une section est caractérisée par sa longueur, son diamètre, sa date d'installation et la liste des réparations qu'elle a subies (date, distance par rapport au nœud amont, type). A l'exception des nœuds terminaux ("racines" et "feuilles"), un nœud relie une section amont (alimentant le nœud) et une ou plusieurs sections aval (alimentées par le nœud), l'ensemble formant une forêt (ensemble d'arbres).

Un nœud comprend une vanne dont on connaît le modèle. La racine d'un arbre est un nœud spécial qui n'est pas connecté à une section amont, mais qui reçoit l'eau d'une installation d'alimentation, identifiée par un numéro, et d'un type tel que "captage", "réservoir", "station d'épuration", etc. Chaque installation d'alimentation alimente un arbre du réseau. Les feuilles d'un arbre n'alimentent pas de sections aval. Chaque nœud est repéré par son adresse; il est caractérisé par sa profondeur. Une section ne fait pas l'objet de plus d'une réparation par jour.

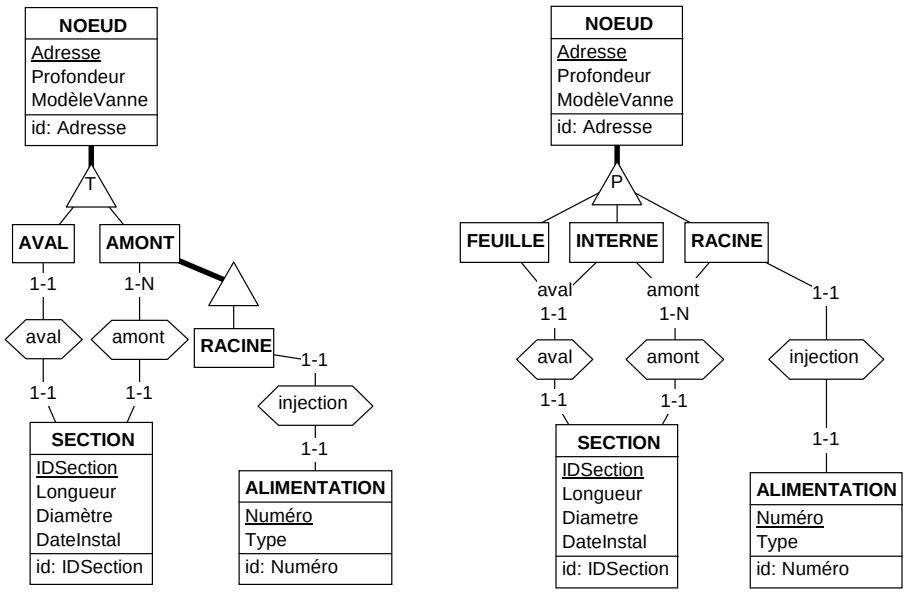
Chaque client (nom, adresse) possède un et un seul compteur identifié par son numéro, et branché sur une section. La position de ce branchement est indiquée par la distance à partir du nœud amont. Pour chaque client, on enregistre les consommations annuelles.

Solution

Première approche simplifiée :



Remarques. (1) SECTION, un concept majeur, ne dispose que d'un identifiant implicite (son NOEUD aval), (2) les différentes sortes de noeuds ne sont pas mises en évidence. D'où les deux variantes ci-dessous. Certaines contraintes sont manquantes.



17.4 Etudier soigneusement les principales pages d'un site de commerce électronique tel que www.amazon.fr, www.fnac.com ou www.ebay.fr. Il apparaît clairement que les informations qu'elles contiennent sont extraites

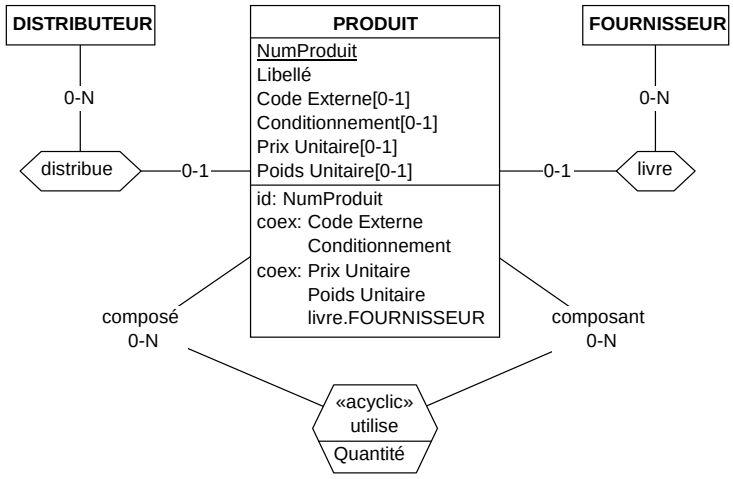
d'une base de données. Identifier dans le site choisi une partie réduite, telle que *Musique / Meilleures ventes* chez Amazon, et proposer un schéma conceptuel des concepts et des informations qui y sont représentés.

17.5 Reprenons le problème consistant à représenter un ensemble de produits et leur composition, déjà rencontré à plusieurs reprises dans cet ouvrage :

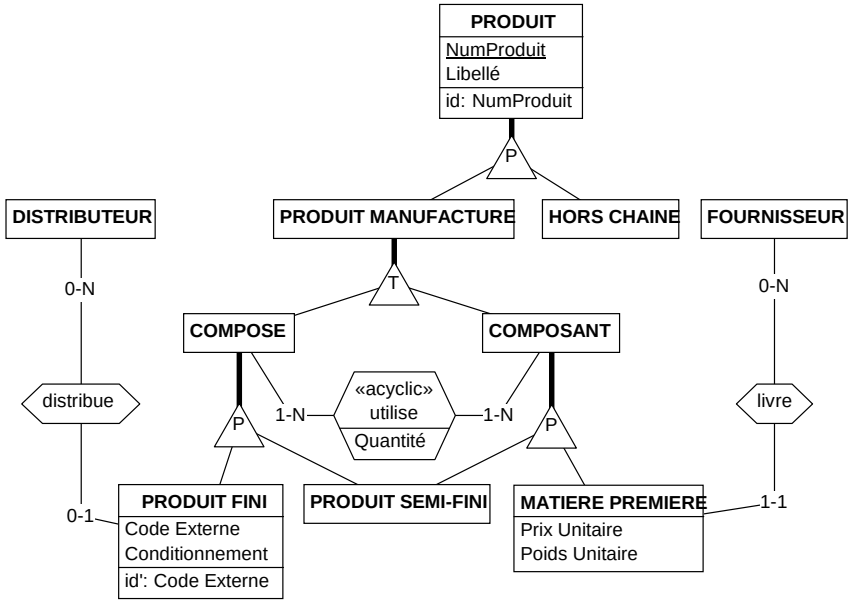
Chaque produit peut être composé de produits plus simples, eux-mêmes éventuellement composés d'autres produits. Pour chaque produit composé, on indique la quantité de chaque composant qui est nécesaire pour en fabriquer une unité. Pour les produits élémentaires (non composés) on connaît le prix unitaire, le poids unitaire et le fournisseur qui peut le livrer. Tout produit a un numéro identifiant et un libellé. Un produit fini possède un code externe identifiant, un conditionnement et éventuellement un distributeur. Un produit n'intervient pas, même indirectement, dans sa propre composition.

Solution

L'énoncé indique que la structure des produits est un *graphe hiérarchique*. Nous proposons d'abord un schéma très simple



Il peut cependant être utile de distinguer les différentes classes de produits, qui sont dotées de propriétés spécifiques. L'énoncé a déjà identifié les produits élémentaires, communément dénommés *matières premières* et les produits finis. Le schéma ci-dessous est une spécialisation du précédent. Il détaille les autres classes de produits et assigne à certaines d'entre elles des propriétés spécifiques, sous la forme d'attributs, de rôles et d'identifiants.



PRODUIT FINI ∩ MATIERE PREMIERE = ∅

17.6 La partie *achat* du site commercial mentionné à la question précédente comporte aussi des informations relatives aux clients, aux paiements et aux conditions de livraison. Il est évident que la société conserve des informations sur les achats précédents des clients, de manière à personnaliser les pages qui sont présentées à un visiteur particulier.

Proposer une extension du schéma de la question précédente qui décrive ces nouvelles informations.

Remarque. Les exercices 17.4 et 17.6 illustrent une problématique qui prend actuellement de plus en plus d'importance : comprendre la structure sémantique de sites étrangers afin d'en extraire de manière intelligente des données pertinentes. Cet exercice relève de la *rétro-ingénierie des site web*.

17.7 Proposer un schéma conceptuel traduisant les concepts de l'énoncé suivant.

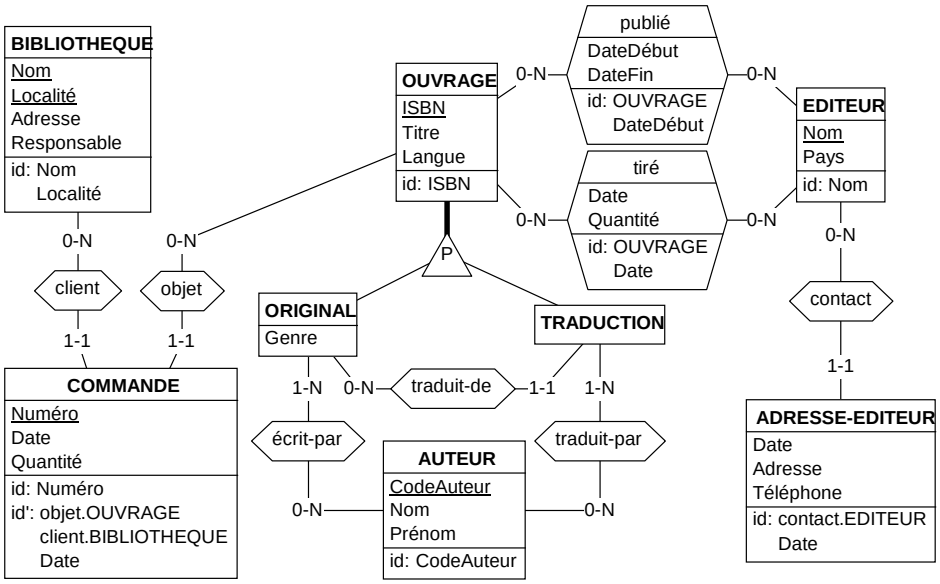
Dans un réseau de bibliothèques, chacune d'elles commande des exemplaires d'ouvrages repris dans une liste de référence. Chaque commande est numérotée (de manière à être identifiable) et reprend la quantité, la date et l'ouvrage concerné. Une bibliothèque n'envoie pas plus d'une commande d'un même ouvrage le même jour.

Un ouvrage peut être publié par un éditeur pendant une certaine période. A tout moment, l'éditeur d'un ouvrage peut céder ses droits de publication à un autre éditeur (il peut même les réacquérir plus tard). D'un ouvrage, un éditeur qui en possède les droits peut tirer, à une certaine date, un nombre déterminé d'exemplaires. Si, après au moins 3 mois, les ventes s'avèrent

meilleures que prévu, l'éditeur effectuera un nouveau tirage (et ainsi de suite). Un ouvrage est caractérisé par un numéro ISBN qui l'identifie, son titre et sa langue. Un ouvrage est soit un original (auquel cas il est caractérisé par un genre), écrit par un ou plusieurs auteurs, soit une traduction d'un original. Dans ce dernier cas, l'ouvrage est traduit par un ou plusieurs auteurs (pour simplifier, vrais auteurs et traducteurs sont considérés comme des auteurs). Une bibliothèque est identifiée par son nom et sa localité. Elle a une adresse et un responsable.

Un éditeur est identifié par son nom et caractérisé par son adresse et son pays d'origine. Cependant, son adresse peut changer au cours du temps. L'adresse est accompagnée d'un numéro de téléphone. Un auteur porte un code unique et est caractérisé par son nom et son prénom.

Solution



17.8 Un organisme gère les activités hôtelière d'un ensemble d'établissements d'une région. Chaque hôtel porte un nom unique, appartient à une catégorie, et est ouvert durant une période déterminée. On enregistre son adresse. Il dispose de chambres, classées par catégories, propres à l'hôtel. Le prix d'une chambre dépend de sa catégorie, qui précise l'équipement disponible : bain, douche et téléviseur. Les chambres d'un hôtel ont des numéros distincts. Elles sont situées à un étage et offrent un certain nombre de lits à une personne et à deux personnes.

Les clients occupent des chambres, avec ou sans réservation. Lors d'une réservation, dont on enregistre le numéro et la date, le client demande une ou plusieurs chambres. Pour chacune, il précise soit une chambre précise soit

une catégorie, ainsi que la période d'occupation prévue. L'hôtelier vérifie s'il dispose de chambres de cette catégorie durant cette période. Par la suite, il pourra affecter une chambre réelle à la réservation selon la catégorie.

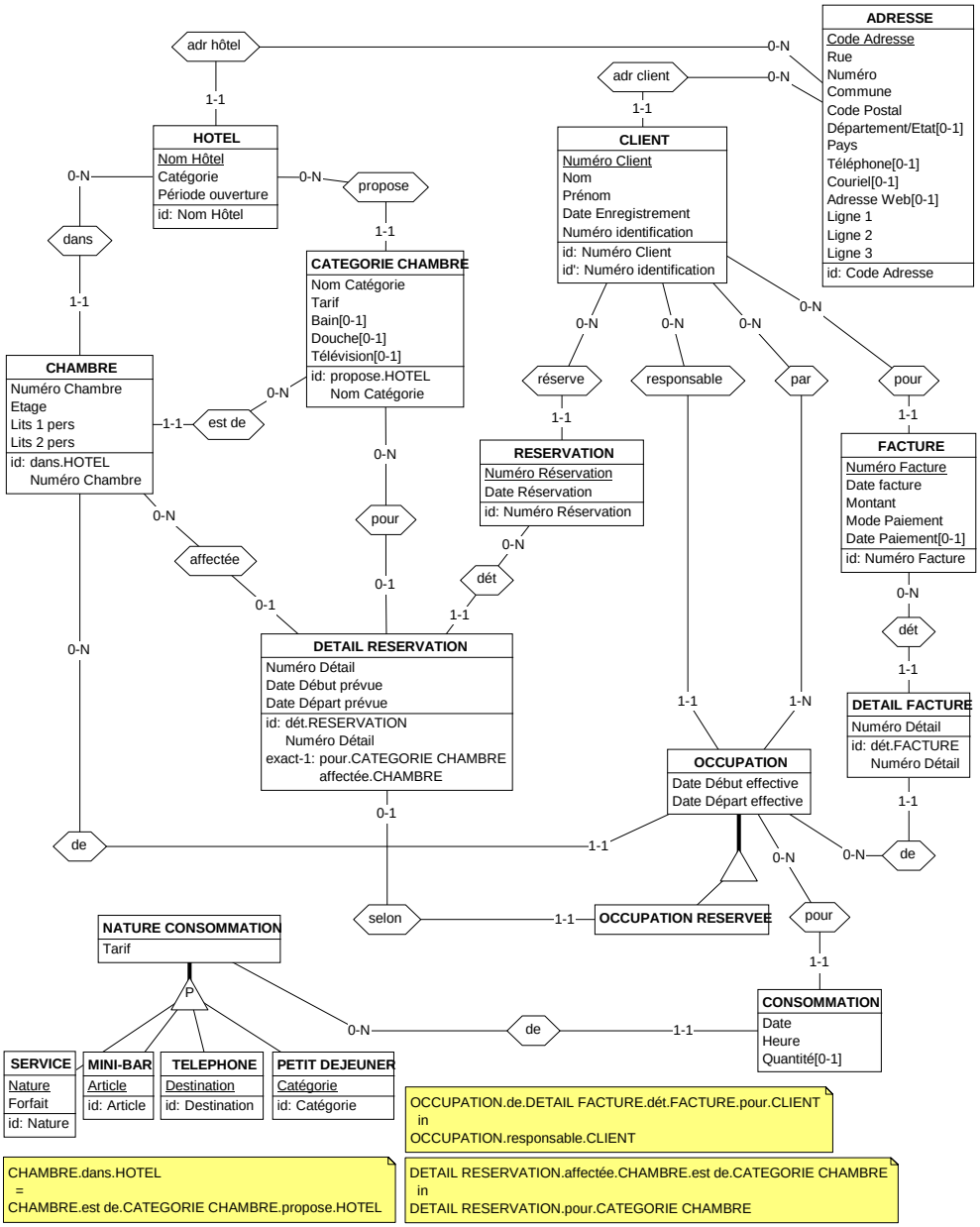
Une chambre est occupée par un ou plusieurs clients durant une certaine période effective. Si cette occupation a fait l'objet d'une réservation, on indique celle-ci.

Les clients occupant une chambre peuvent consommer des services (blanchisserie, sauna, etc.), des articles du mini-bar, des communications téléphoniques (dont le tarif dépend de la destination) et des petits déjeûners (il existe plusieurs catégories, de prix différents). On connaît le tarif de chaque type de consommation; pour un service on indique en outre si le prix est forfaitaire ou proportionnel. Pour chaque consommation, on connaît la date, l'heure et, si pertinent, la quantité. Le responsable de l'occupation d'une chambre est le client qui assumera le paiement; il n'occupe pas nécessairement cette chambre. Lors de l'enregistrement d'un client, on lui affecte un numéro de client et on note son nom, son prénom et son adresse.

Une adresse est composée d'un nom de rue, d'un numéro d'immeuble, du nom de la commune et du code postal, du département ou de l'état (si pertinent), du pays, et, s'ils existent, d'un numéro de téléphone, d'une adresse de courriel et d'une adresse web. On enregistre aussi la même adresse sous la forme d'une à trois lignes qui sont à imprimer telles quelles sur les enveloppes. Plusieurs clients peuvent partager la même adresse.

A la date du départ, une facture est établie et remise au client responsable des occupations. La facture est numérotée et reprend en outre le mode de paiement et, lorsque celui-ci est effectué, sa date. La facture détaille les chambres occupées ainsi que les consommations de chacune. Le montant de la facture peut être différent de la somme des chambres occupées et des consommations (suite à une remise par exemple).

Solution



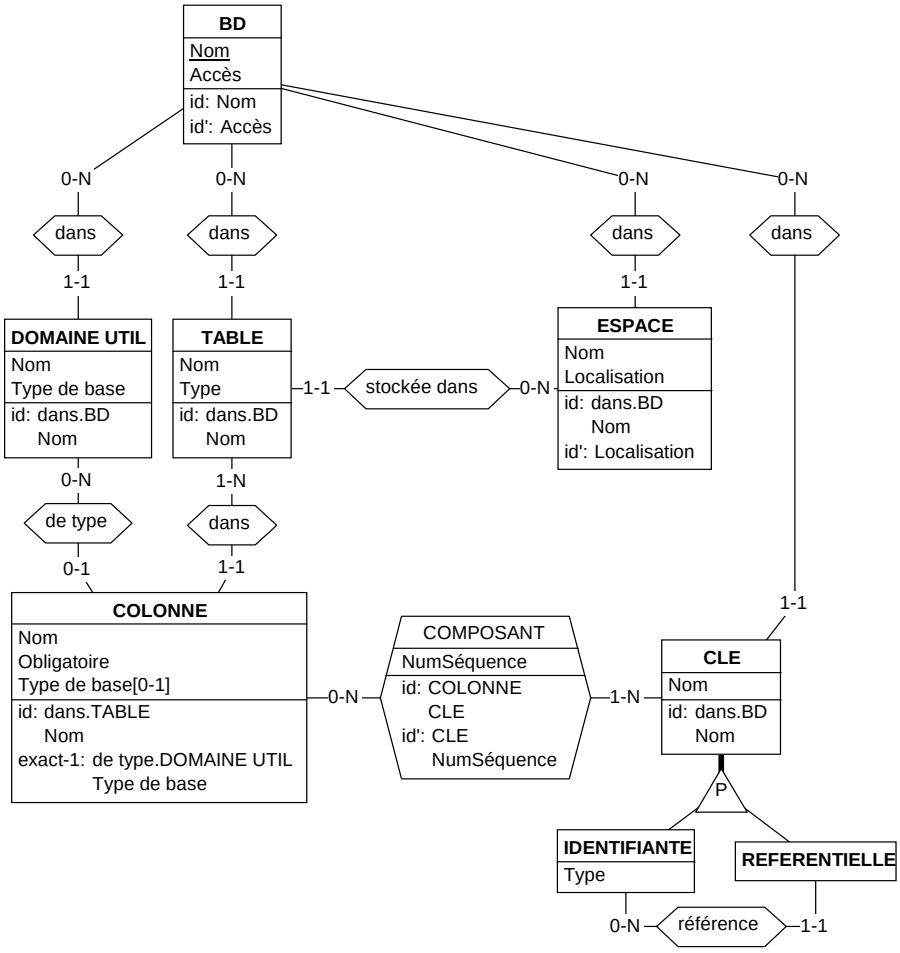
17.9 Proposer un méta-schéma pour le modèle SQL2 (exprimé dans le modèle Entité-associations étendu). Plus précisément, on désire contruire un dictionnaire de données qui représente les structures d'une collection de bases de données. Chaque base de données porte un nom identifiant et est localisée par son chemin d'accès (URL).

Chaque base de données est constituée d'un nombre quelconque de tables, elles-mêmes constituées chacune d'au moins une colonne. Une table est réelle (de base) ou virtuelle (vue). Les lignes d'une table sont stockées dans un espace de stockage de la base de données. Les règles d'unicité des noms des tables, des colonnes et des espaces sont celles d'SQL. La localisation d'un espace de stockage est unique. Une colonne est obligatoire ou facultative. Elle est définie sur un type de base (*decimal*, *character*, *bit*, etc.) ou sur un domaine défini par l'utilisateur. Ce dernier porte un nom unique dans la base de données et est défini sur un type de base.

Un certain nombre de clés peuvent être attachées à une table. Une clé est identifiante (auquel cas elle est primaire ou secondaire) ou référentielle (clé étrangère). Elle porte un nom unique dans la base de données. Une clé est composée d'une séquence d'au moins une colonne de sa table. Une colonne ne peut apparaître plus d'une fois dans une même clé. A chaque clé référentielle est associée une clé identifiante de même composition.

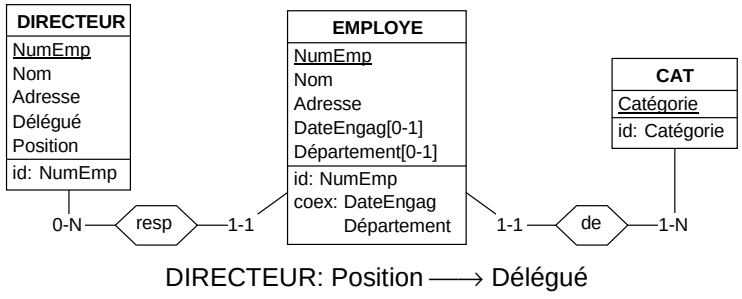
Solution

On propose le schéma ci-dessous. Il devrait être complété de diverses contraintes d'intégrité. On pourrait aussi le compléter par la représentation des *View*, des *index*, des *triggers*, des *checks* et des *privilèges*.



17.10 Normaliser le schéma de la figure 12.26 à la lumière des concepts développés dans ce chapitre.

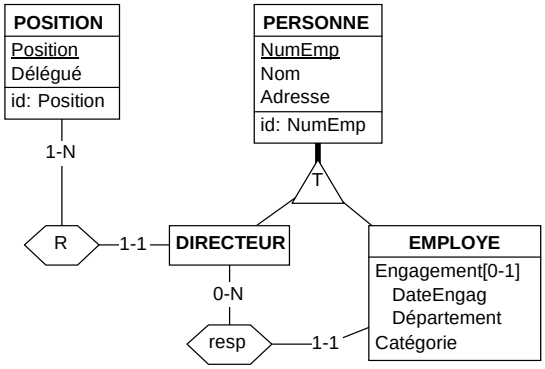
17.11 Normaliser le schéma ci-dessous.



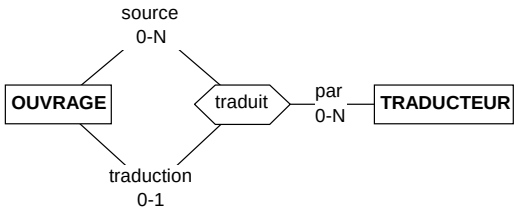
Solution

Ce schéma comporte quatre sous-structures non normalisées.

- DIRECTEUR et EMPLOYE ont des attributs commun. On les extrait pour former le surtype PERSONNE. On fait l'hypothèse que l'identifiant NumEmp s'applique à l'union des populations des sous-types.
- La contrainte coex sur les attributs d'EMPLOYE s'exprime de manière plus claire sous la forme d'un attribut composé facultatif.
- Le type d'entités CAT apparaît comme un *type d'entités - attribut*.
- Le déterminant de la DF explicite Position ⟶ Délégué n'est pas un identifiant de DIRECTEUR. On extrait les attribut litigieux Position et Délégué sous la forme d'un type d'entités autonome.

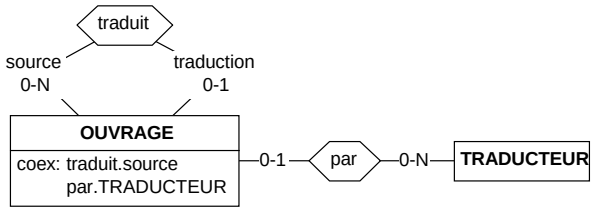


17.12 Normaliser le schéma ci-dessous, qui exprime le fait que certains ouvrages ont été traduits par des traducteurs.



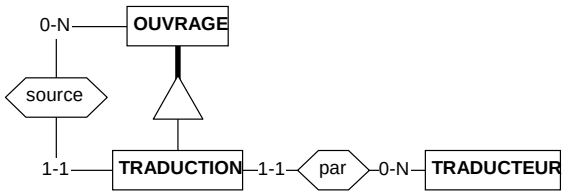
Solution

Ce schéma n'est pas normalisé, car il comporte un type d'associations ternaire dont un rôle est de cardinalité [0-1]. Cette propriété n'entraînant pas de problème de redondance, il est possible de conserver ce schéma tel quel. Si on désire pousser plus loin la normalisation, on décomposera le type d'associations, ce qui conduit au schéma suivant :



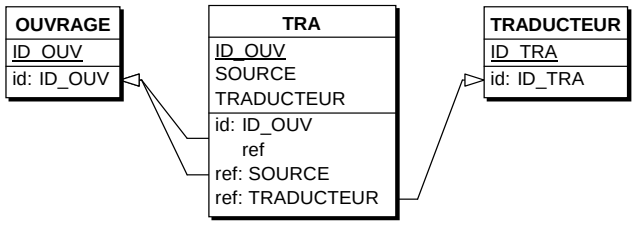
Ce schéma présente deux défauts. D'une part, il introduit une contrainte additionnelle (*coexistence*), et d'autre part il comporte deux rôles facultatifs susceptibles d'entraîner des problèmes dans une implémentation relationnelle (clés étrangères à valeur null).

La contrainte de coexistence peut s'exprimer de manière structurelle via le sous-type TRADUCTION, qui met en évidence le concept d'*ouvrage traduit*. Ce schéma sera sans doute plus évolutif que les précédents si ce concept devait prendre de l'importance dans la suite. En outre, il ne contient plus les rôles facultatifs problématiques.



Discussion additionnelle

Il est intéressant d'observer que le premier et le troisième schéma se traduisent, selon le plan de transformation classique, par des schémas relationnels apparemment identiques (Figure 8.121).

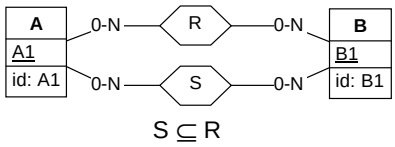


En fait, ces schémas diffèrent par l'interprétation de la table TRA (dont le nom a expressément été choisi pour sa neutralité : TRADUIT ou TRADUCTION). Dans le premier cas, cette table représente les opérations de traduction, les ouvrages traduits étant représentés exclusivement dans la table OUVRAGE. Dans le deuxième cas, la table TRA représente les ouvrages traduits. Un tel ouvrage est représenté par un fragment extrait de TRA auquel on joint le

fragment correspondant dans la table OUVRAGE. Si, dans le futur, il était nécessaire d'ajouter des propriétés spécifiques aux ouvrages traduits, il faudrait les ajouter à la table OUVRAGE dans le premier cas, sous forme de composants facultatifs, et à la table TRA dans le second.

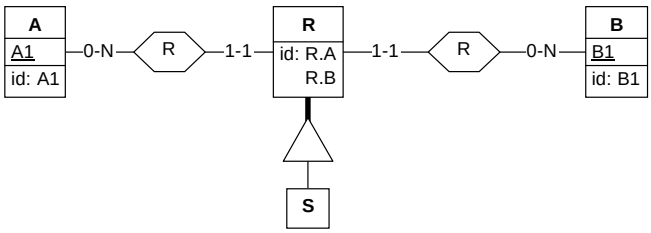
Dans le modèle relationnel objet, qui supporte les relations *is-a*, le différence serait bien entendu nettement marquée, et cette discussion serait sans objet.

17.13 Eliminer la contrainte d'inclusion ci-dessous en la représentant de manière structurelle.

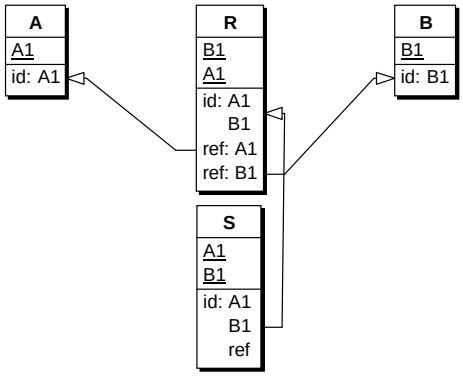


$S \subseteq R$

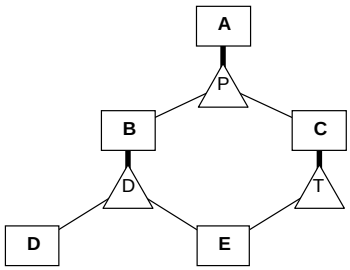
La construction la plus fréquente qui soit basée sur une contrainte d'inclusion d'extensions est la relation *is-a*. En transformant les deux types d'associations en types d'entités, la contrainte d'inclusion se traduit par une relation *is-a* entre ces derniers. On obtient ainsi le schéma ci-dessous.



Il est également possible de simplifier ce schéma selon le procédé de la section 17.5.1/e. On notera que le schéma relationnel qui dérive naturellement de ce schéma normalisé est simple et intuitif :

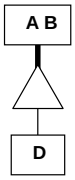


17.14 En utilisant le concept de *satisfiabilité*, simplifier le schéma ci-dessous

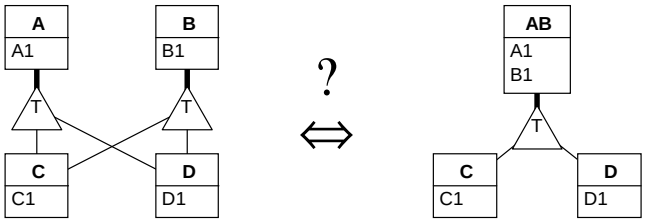


Solution

En raison de la contrainte **P** sur A, le type d'entités E est non satisfiable. Les populations de C et E sont les mêmes (contrainte **T** sur C). C est donc lui aussi non satisfiable. Par conséquent les populations de A et B sont les mêmes. On peut donc fusionner A et B en AB, qui regroupe leurs attributs, rôles et contraintes. Il vient donc :



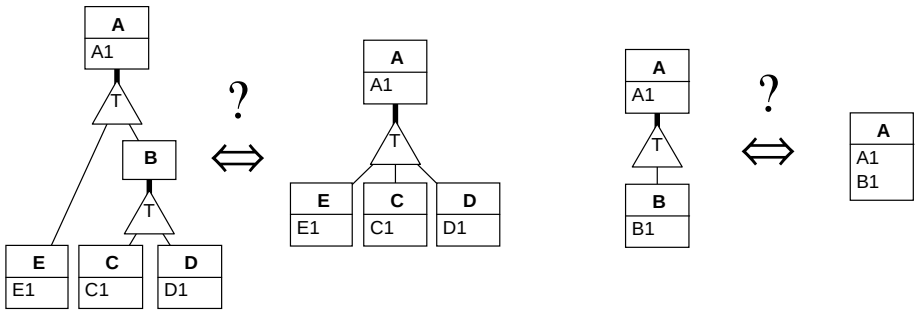
17.15 Il est possible de prouver l'équivalence de deux types d'entités en démontrant qu'ils ont la même population. Montrons-le dans l'exemple ci-dessous (on admet que A et B ont un surtype commun) :



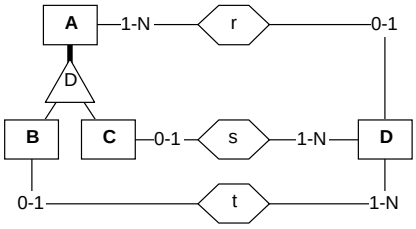
On a : $pop(A) = pop(C) \cup pop(D)$; $pop(B) = pop(C) \cup pop(D)$

d'où : $pop(A) = pop(B)$

On peut donc admettre que les types d'entités A et B sont identiques. On les renomme AB. Déterminer de la même manière si, dans chacun des deux couples ci-dessous, les schémas sont équivalents.



17.16 Tous les objets du schéma ci-dessous sont-ils satisfiables (d'après [Vigna, 2004]) ?



Solution

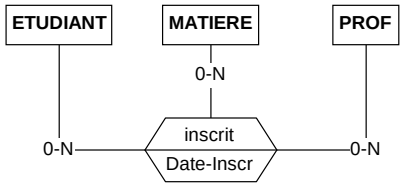
de r on tire : $\mathbf{N}_D \geq \mathbf{N}_A$ de s on tire : $\mathbf{N}_C \geq \mathbf{N}_D$

de t on tire : $\mathbf{N}_B \geq \mathbf{N}_D$ de la relation *is-a* on tire : $\mathbf{N}_A \geq \mathbf{N}_B + \mathbf{N}_C \geq 2 \times \mathbf{N}_D$

on a donc : $\mathbf{N}_A \geq 2 \times \mathbf{N}_A$, ce qui n'est valide que si $\mathbf{N}_A = 0$.

On en conclut qu'aucun des quatre types d'entités n'est satisfiable.

17.17 Normaliser, si nécessaire, le type d'associations ci-dessous[6].



inscrit: MATIERE ⟶ PROFESSEUR

Solution

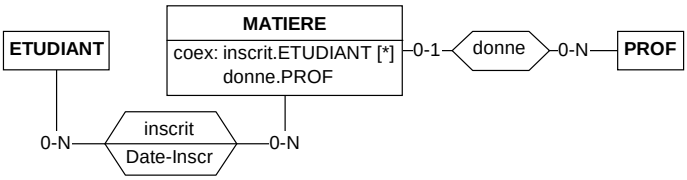
La sémantique interne du type d'associations inscrit est la suivante :

inscrit(ETUDIANT, MATIERE, PROF, Date-Inscr)

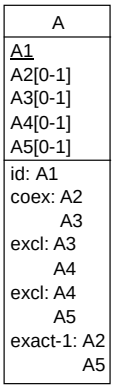
6. Attention, les règles relatives aux identifiants implicites sont de stricte application. En particulier, on tiendra compte de l'identifiant implicite avant de faire intervenir la dépendance fonctionnelle.

inscrit: MATIERE ⟶ PROFESSEUR

La décomposition de ce schéma sous 3e forme normale correspond au schéma ci-dessous. La contrainte de coexistence disparaît si on admet que *toute matière est dispensée par un professeur.*



17.18 Vérifier la cohérence du schéma ci-dessous et normaliser celui-ci si nécessaire.



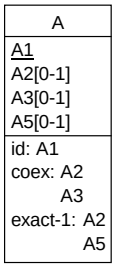
Est-il possible qu'une entité A respecte simultanément les 4 contraintes ? Si oui, alors le schéma est cohérent. Il existe plusieurs manières d'aborder cette question. En voici une première qui s'inspire de l'évaluation d'une fonction booléenne complexe par calcul de sa table de vérité.

Pour toute entité A, chacun des attributs A2, A3, A4, A5 peut avoir une valeur (ce qu'on note 1) ou non (noté 0). Il existe donc 24 = 16 configurations possibles. Pour chacune d'elles, on évalue chacune des contraintes (VRAI ou FAUX). Les configurations pour lesquelles les 4 contraintes ont la valeur VRAI sont cohérentes. Les autres ne peuvent se produire. Le tableau de calcul est le suivant :

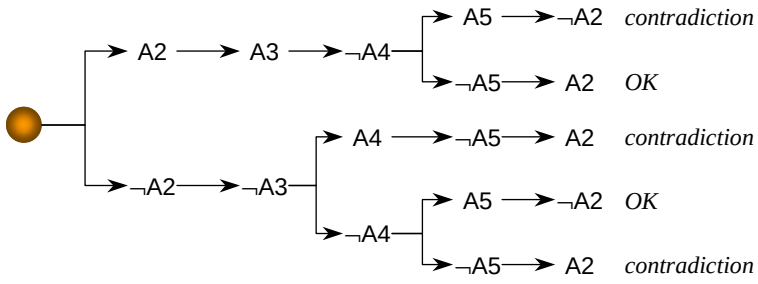
A2	A3	A4	A5	coex:A2,A3	excl:A3,A4	excl:A4,A5	exact1:A2,A5	total
0	0	0	0	VRAI	VRAI	VRAI	FAUX	FAUX
0	0	0	1	VRAI	VRAI	VRAI	VRAI	VRAI
0	0	1	0	VRAI	VRAI	VRAI	FAUX	FAUX
0	0	1	1	VRAI	VRAI	FAUX	VRAI	FAUX

0	1	0	0	FAUX	VRAI	VRAI	FAUX	FAUX
0	1	0	1	FAUX	VRAI	VRAI	VRAI	FAUX
0	1	1	0	FAUX	FAUX	VRAI	FAUX	FAUX
0	1	1	1	FAUX	FAUX	FAUX	VRAI	FAUX
1	0	0	0	FAUX	VRAI	VRAI	VRAI	FAUX
1	0	0	1	FAUX	VRAI	VRAI	FAUX	FAUX
1	0	1	0	FAUX	VRAI	VRAI	VRAI	FAUX
1	0	1	1	FAUX	VRAI	FAUX	FAUX	FAUX
1	1	0	0	VRAI	VRAI	VRAI	VRAI	VRAI
1	1	0	1	VRAI	VRAI	VRAI	FAUX	FAUX
1	1	1	0	VRAI	FAUX	VRAI	VRAI	FAUX
1	1	1	1	VRAI	FAUX	FAUX	FAUX	FAUX

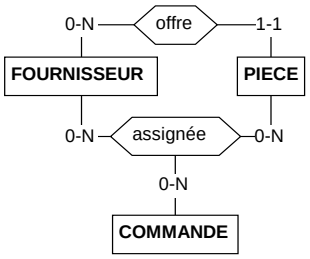
On observe que 2 configurations satisfont simultanément des 4 contraintes. Le schéma est donc cohérent. En revanche, toutes les autres configurations d'attributs sont invalides et n'apparaîtront jamais. On observe en outre que l'attribut A4 n'a jamais de valeur dans les deux configurations valides et qu'il peut donc être supprimé. Le schéma se simplifie comme suit :



On peut proposer une autre méthode d'analyse, plus simple et plus rapide. On construit un arbre définissant, pour la suite des attributs A2, A3, A4, A5, leur état correspondant aux contraintes. On observe les contradictions, selon lesquelles on déduit d'un attribut qui a une valeur le fait qu'il n'en a pas, ou le contraire. Partant de A2, qui soit a une valeur (branche supérieure, notée "A2") soit n'en a pas de valeur (branche inférieure, notée "¬A2"), l'arbre se présente comme ci-dessous. Deux configurations sur les 5 ne présentent pas de contradictions ("OK"). Elles se caractérisent par l'absence de valeur pour A4. On peut alors simplifier le schéma comme ci-dessus.



17.19 Le schéma ci-dessous comporte des constructions contradictoires. Il peut aussi être décrit comme un schéma non normalisé. Proposer une version normalisée et cohérente.



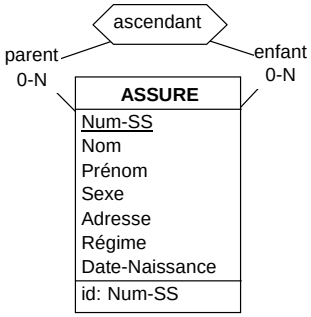
assignée[FOURNISSEUR,PIECE] ⊆ offre[FOURNISSEUR,PIECE]

Solution

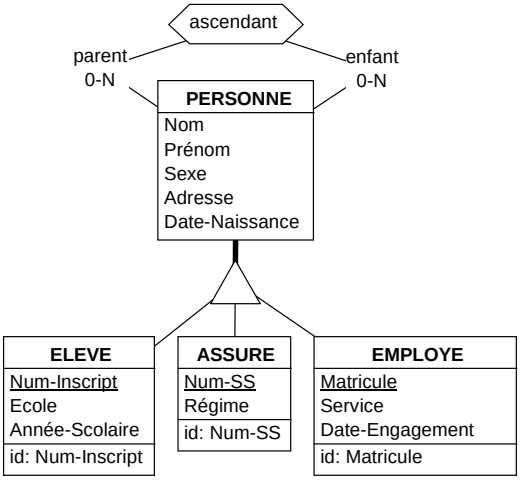
17.20 Proposer un schéma conceptuel minimal intégrant les informations contenues dans les trois types d'entités ci-dessous.

ELEVE
Num-Inscript
Nom
Adresse
Prénom
Date-Naissance
Ecole
Année-Scolaire
Père[0-1]
Mère[0-1]
id: Num-Inscript

EMPLOYE
Matricule
Nom
Prénom
Adresse
Service
Date-Engagement
Date-Naissance
Prénom-Enfant[0-10]
id: Matricule



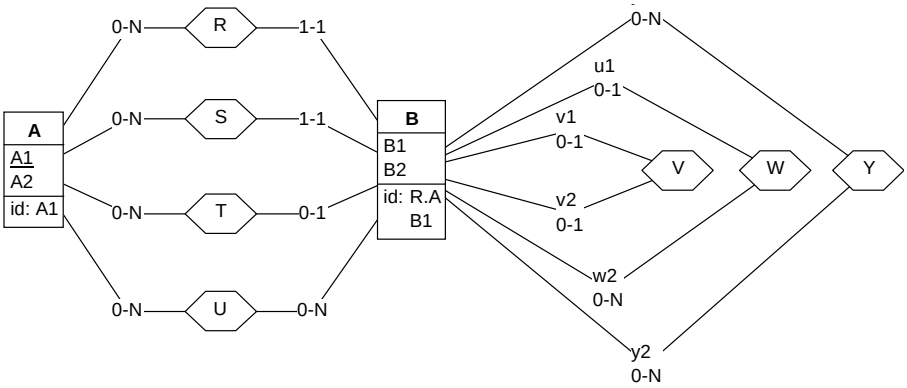
Solution


ascendant
parent
0-N
enfant
0-N
PERSONNE
Nom
Prénom
Sexe
Adresse
Date-Naissance
ELEVE
Num-Inscript
Ecole
Année-Scolaire
id: Num-Inscript
ASSURE
Num-SS
Régime
id: Num-SS
EMPLOYE
Matricule
Service
Date-Engagement
id: Matricule

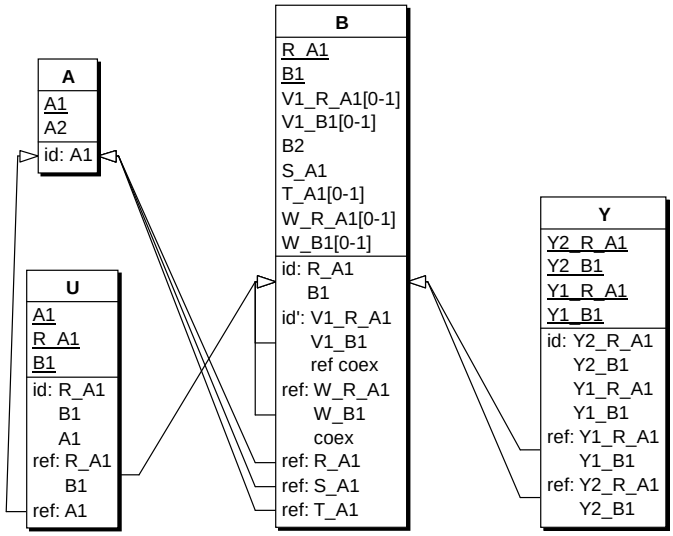
A.18 CHAPITRE 18 - CONCEPTION LOGIQUE D'UNE BASE DE DONNÉES RELATIONNELLE

18.1 Reprendre les exercices du chapitre 14 et les résoudre en appliquant le plan de transformation développé dans ce chapitre. Pour ce qui est de l'exercice 13.2, on ajoutera la propriété suivante au schéma conceptuel : *le cheval senior d'une écurie appartient à celle-ci.*

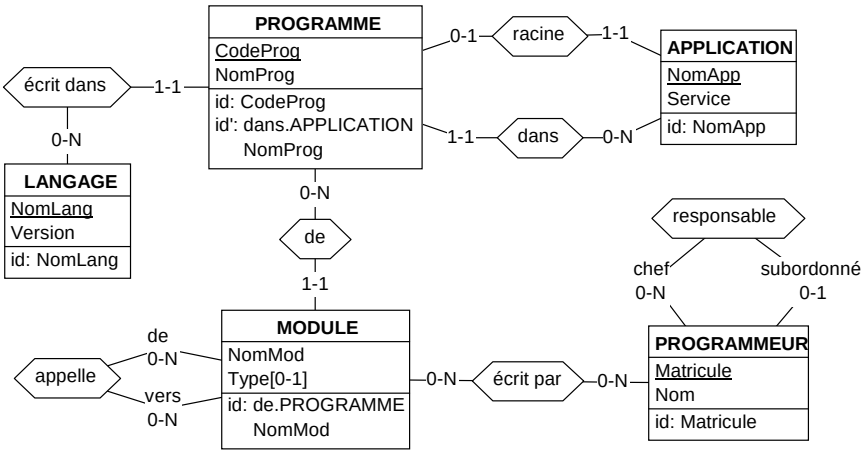
18.2 Traduire le schéma conceptuel ci-dessous en structure de tables.



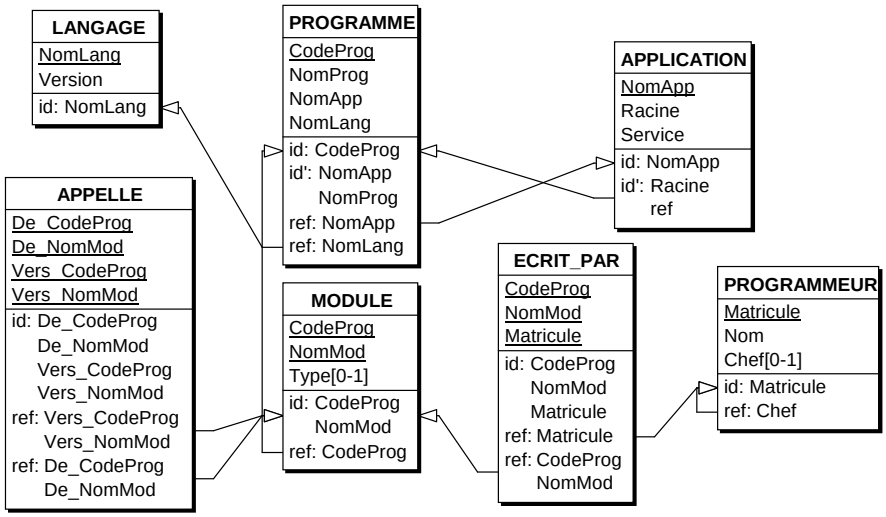
Solution



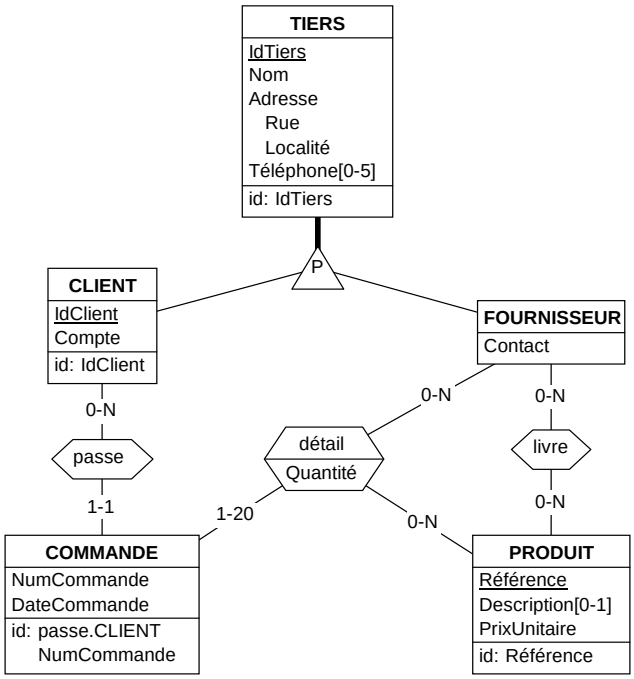
18.3 Traduire le schéma conceptuel ci-dessous en structure de tables.



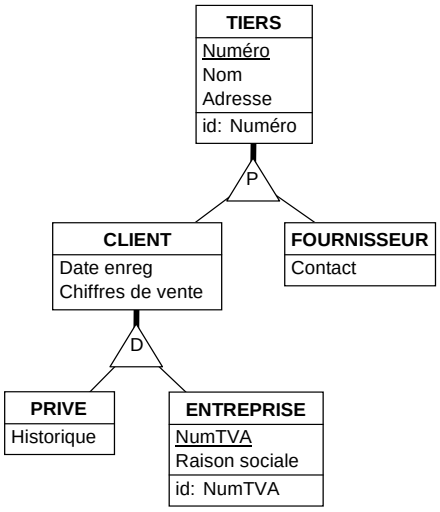
Solution



18.4 Traduire le schéma conceptuel ci-dessous en structure de tables.

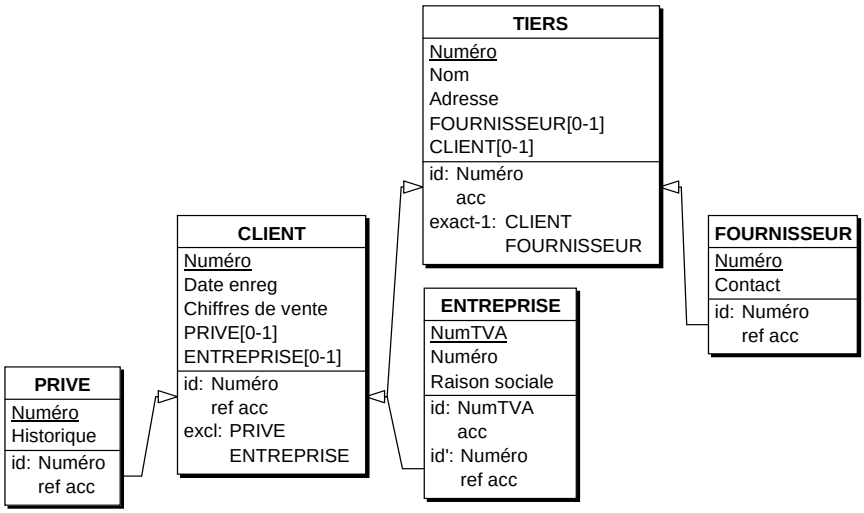


18.5 Transformer le schéma ci-dessous en structures relationnelles selon l'approche standard (matérialisation). Produire ensuite, en utilisant les transformations par héritage descendant et héritage descendant, un schéma ne comportant qu'une seule table (TIERS), un schéma ne comportant que deux tables (CLIENT et FOURNISSEUR) puis un schéma constitué de la représentation exclusive des sous-types. Comparer ces schémas.

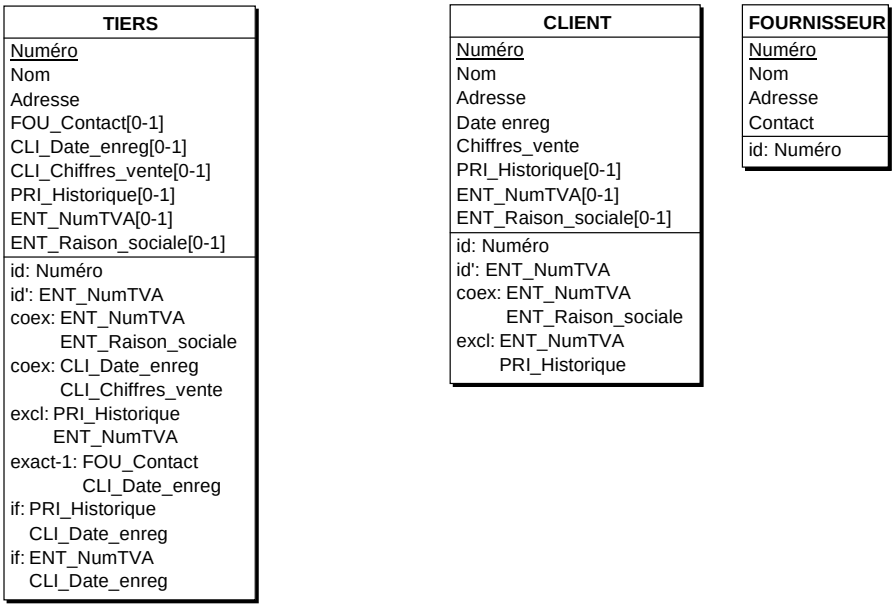


Solution

1. Approche standard (matérialisation) :



2. Représentation à 1 tables (gauche) et à 2 tables (droite) :



3. Représentation exlusive des sous-types :

TIERS_CLIENT_PRIVE
Numéro
Nom
Adresse
Date enreg
Chiffres de vente
Historique
id: Numéro

TIERS_CLIENT_ENTREPRISE
Numéro
Nom
Adresse
Date enreg
Chiffres de vente
NumTVA
Raison sociale
id: Numéro
id': NumTVA

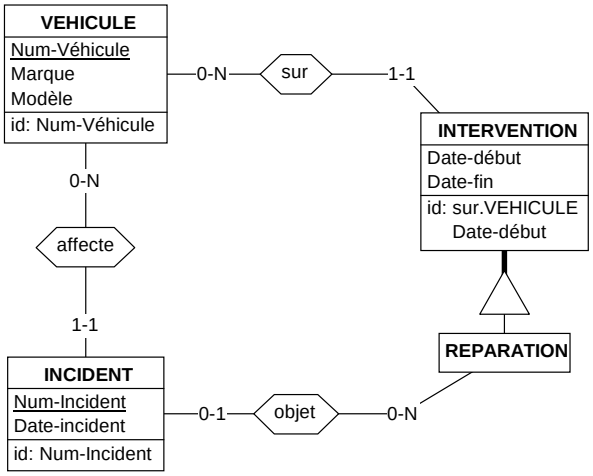
TIERS_CLIENT_AUTRES
Numéro
Nom
Adresse
Date enreg
Chiffres de vente
id: Numéro

TIERS_FOURNISSEUR
Numéro
Nom
Adresse
Contact
id: Numéro

L'expression des contraintes d'intégrité additionnelles dans ces schémas est laissée à l'initiative du lecteur.

18.6 Transformer le schéma suivant en schéma relationnel.

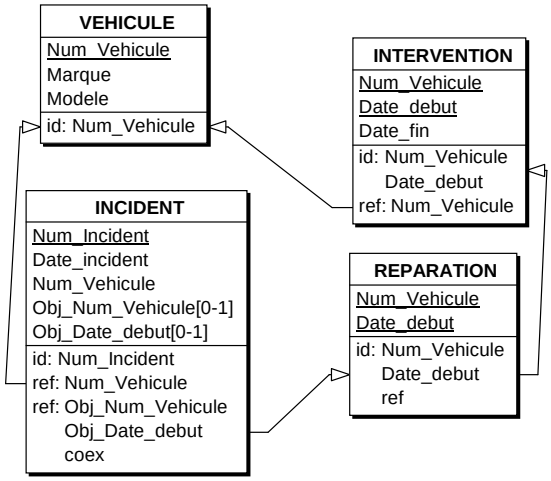
Note : ce problème, en apparence banal, est plus complexe qu'il n'y paraît. Il illustre la difficulté de construire un plan de transformation qui fonctionne dans tous les cas.



INCIDENT.objet.REPARATION.sur.VEHICULE ⊆ INCIDENT.affecte.VEHICULE

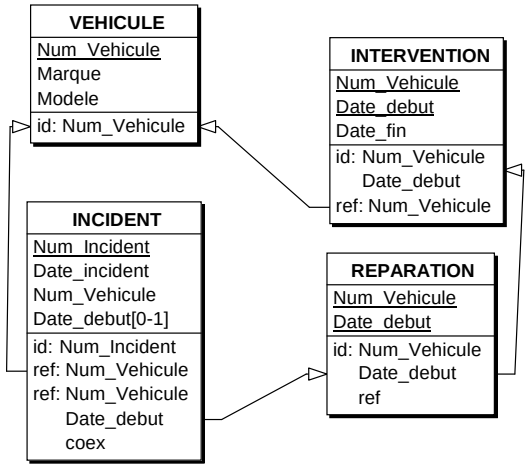
Solution

L'application stricte des règles de la conception logique relationelle conduit au schéma suivant. La contrainte d'égalité des véhicules se traduit par une contrainte d'égalité des colonnes Num_Vehicule et Obj_Num_vehicule de la table INCIDENT.



INCIDENT.Num_vehicule = INCIDENT.Obj_Num_vehicule

L'utilisation d'une seule colonne permet de supprimer cette dernière contrainte. On obtient alors le schéma simplifié ci-dessous.

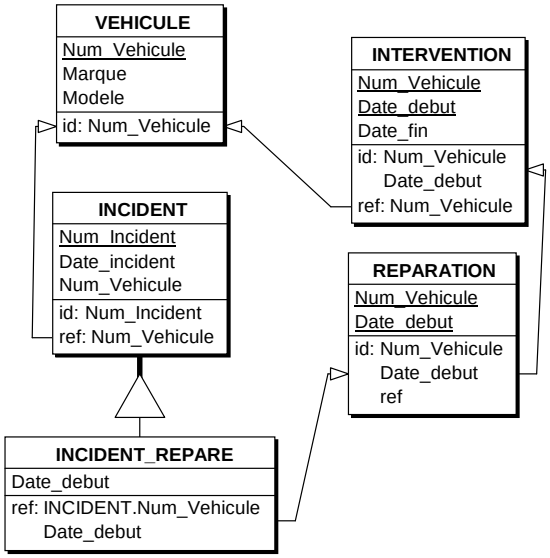


Ce schéma pose deux problèmes :

- la contrainte de coexistence est invalide, un de ses composants étant obligatoire;
- une clé étrangère est semi-facultative, constituée d'un composant obligatoire et d'un composant facultatif; cette construction est invalide dans le modèle relationnel de base tel que nous l'avons défini.

On résout le premier problème par la suppression de la contrainte de coexistence. Le second problème est plus complexe à traiter.

Dans un premier temps, on identifie une classe spécifique d'incidents : les *incidents réparés*, dont la particularité est d'être doté d'un attribut Date_debut obligatoire. C'est la table INCIDENT_REPARE de cette dernière classe qui référence REPARATION, via une clé étrangère constituée de l'attribut hérité Num-Vehicule et de l'attribut propre Date_debut.

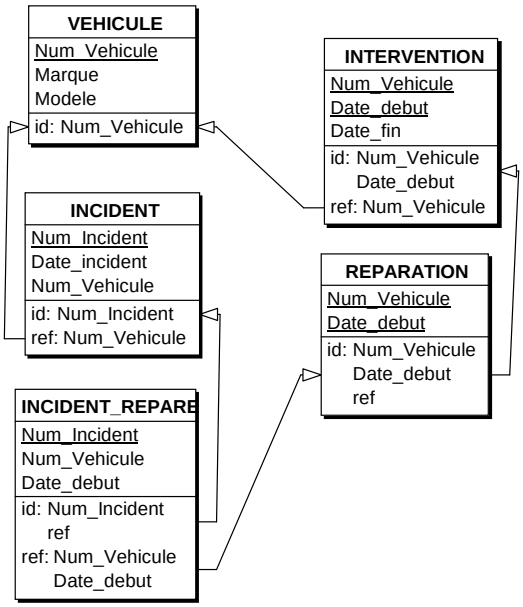


Il nous reste à transformer la relation *is-a*. Nous la transformons d'abord en type d'associations 1:1. Malheureusement, ce dernier ne peut à son tour être traduit sous la forme d'une clé étrangère de manière classique[7] : l'identifiant de INCIDENT_REPARE comporte un composant hérité qui ne lui appartient pas en propre. Il faudrait dès alors ajouter à cette table une colonne qui serait une copie de INCIDENT.Num_Véhicule et qui serait assortie d'une contrainte de redondance. On obtiendrait ainsi le schéma ci-après.

Il n'est pas certain que ce schéma soit plus clair que le premier, obtenu par la méthode classique. Tous deux contiennent une redondance et une contrainte qui exprime celle-ci. Dans la dernière solution, le schéma comporte une table supplémentaire (ce qui n'est pas un défaut en soi), une colonne héritée et une contrainte de redondance.

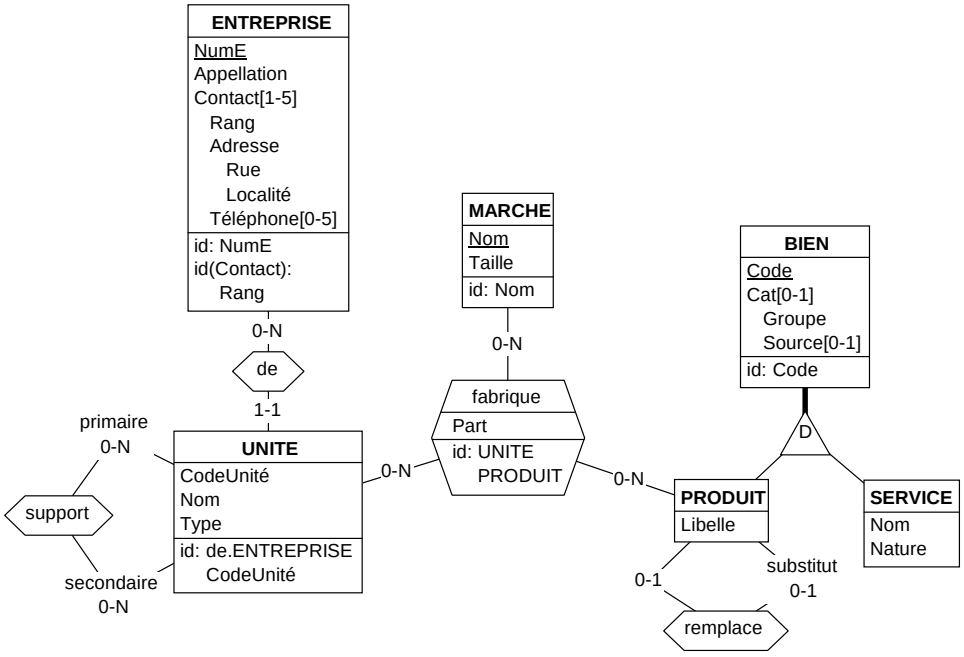
Conclusion : rien n'est parfait en ce bas monde.

7. En SQL2 du moins. En SQL3, le schéma peut être conservé tel quel.

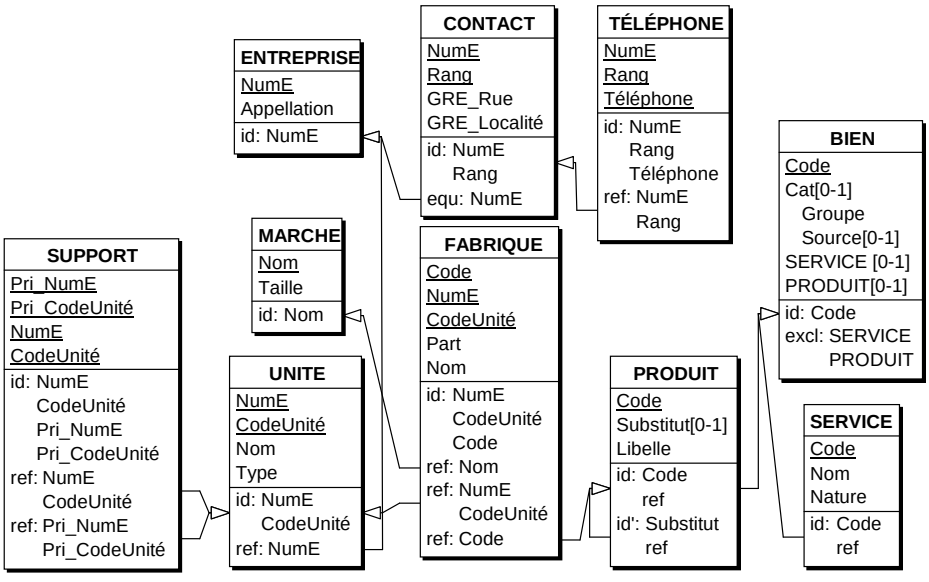


$$\forall ir \in \text{INCIDENT\_REPARE}, \forall i \in \text{INCIDENT},$$
$$ir.\text{Num\_Incident} = i.\text{Num\_Incident} \Rightarrow ir.\text{Num\_vehicule} = i.\text{Num\_vehicule}$$

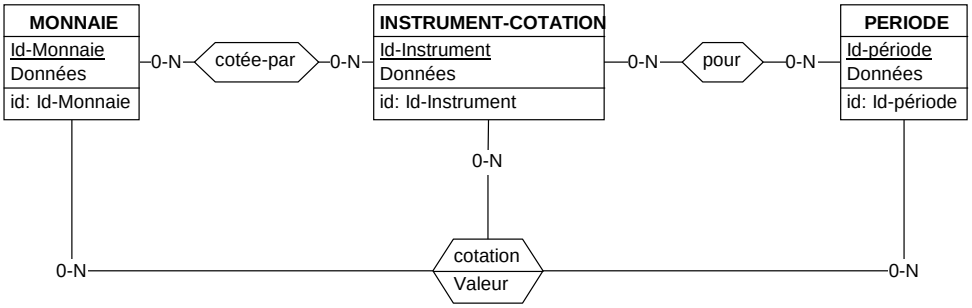
18.7 Transformer le schéma suivant en schéma relationnel.



Solution



18.8 Transformer le schéma suivant en schéma relationnel.

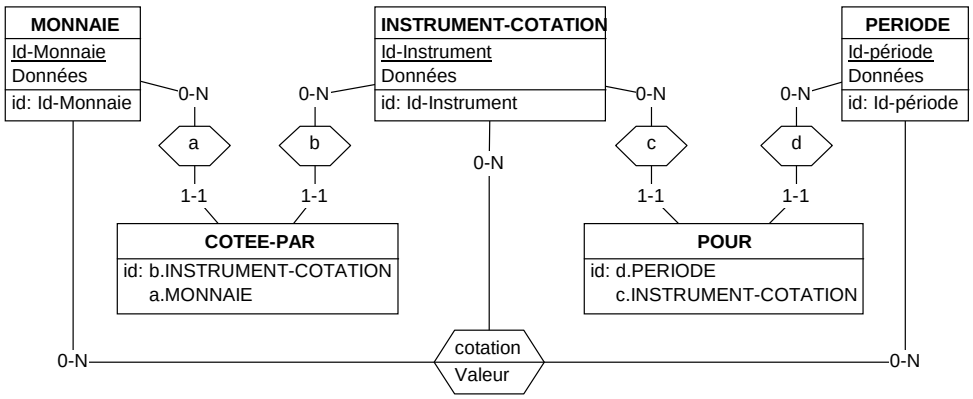


cotation[MONNAIE, INSTRUMENT-COTATION] $\subseteq$ cotée-par

cotation[INSTRUMENT-COTATION, PERIODE] $\subseteq$ pour

Solution

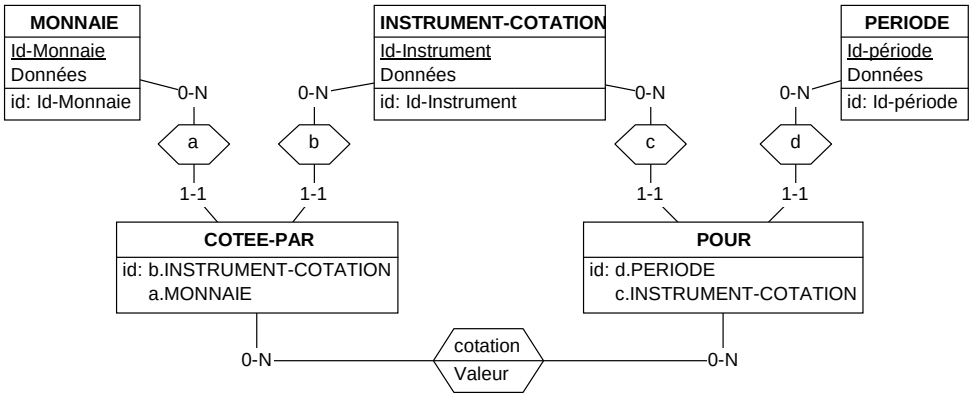
On transforme d'abord les types d'associations cotée-par et pour en types d'entités.



cotation[MONNAIE, INSTRUMENT-COTATION] $\subseteq$ a*b[MONNAIE, INSTRUMENT-COTATION]

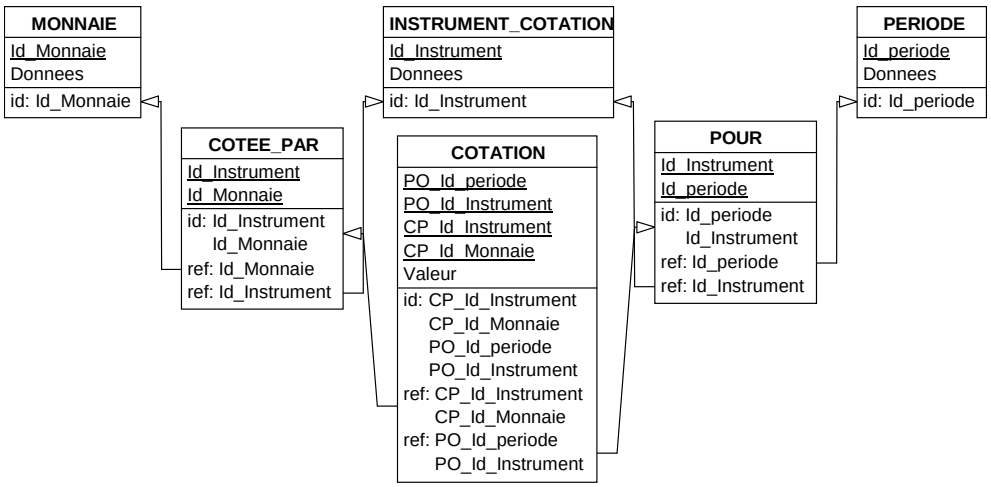
cotation[INSTRUMENT-COTATION, PERIODE] $\subseteq$ c*d[INSTRUMENT-COTATION, PERIODE]

Le type d'entités COTEE-PAR représentant les associations cotée-par (et de même pour POUR), on peut restructurer ce schéma comme ci-dessous. La contrainte exprime le fait que, pour une instance de cotation, les deux instances de INSTRUMENT-COTATION accessibles via COTEE-PAR et POUR sont identiques.



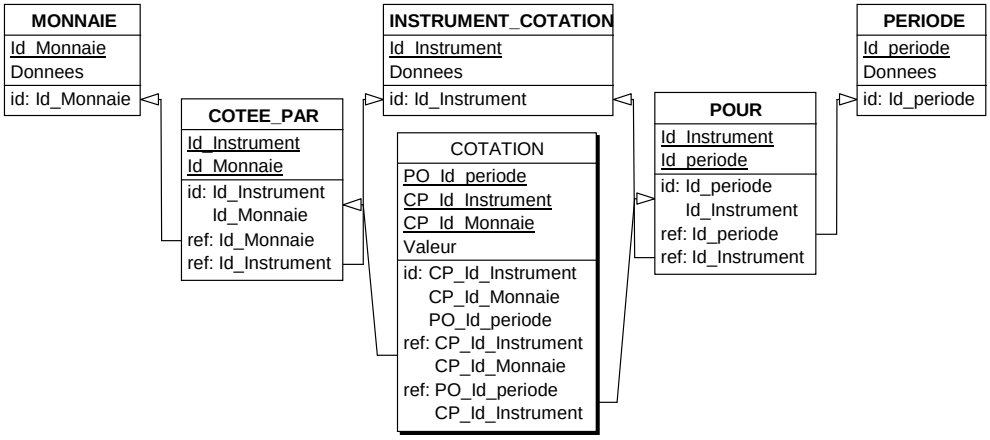
$\forall$(cp,p) $\in$ cotation[COTEE-PAR, POUR],
COTEE-PAR=cp)[INSTRUMENT-COTATION] = c(POUR=p)[INSTRUMENT-COTATION]

La traduction s'achève par la transformation du type d'associations cotation. La contrainte s'exprime par l'égalité des valeurs de deux colonnes.



$\forall$c $\in$ COTATION, c.CP_Id_Instrument = c.PO_Id_Instrument

En réduisant ces deux colonnes en une seule, on obtient le schéma final.



18.9 Développer un plan de transformation pour les diagrammes de classes UML.

18.10 Compléter le plan de transformation de la figure 19.29 de sorte qu'il traite les rôles multi-types.

18.11 Construisez un script DB-MAIN qui traduit le plan de transformation de la question 18.10 . Ce script est-il idempotent ?

A.19 CHAPITRE 19 - CONCEPTION LOGIQUE D'UNE BASE DE DONNÉES RELATIONNELLE OBJET

19.1 Proposer un schéma relationnel objet pour le schéma conceptuel de la figure 1.8.

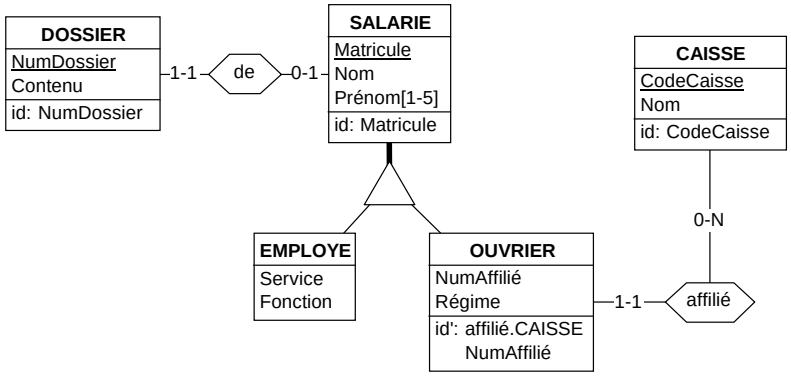
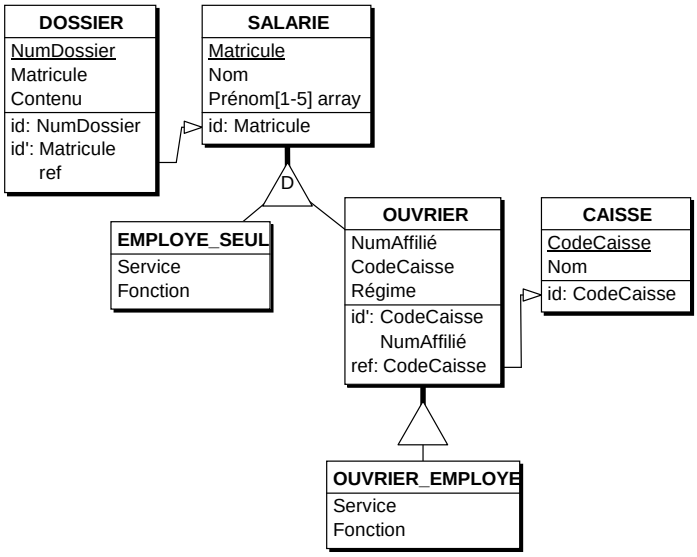


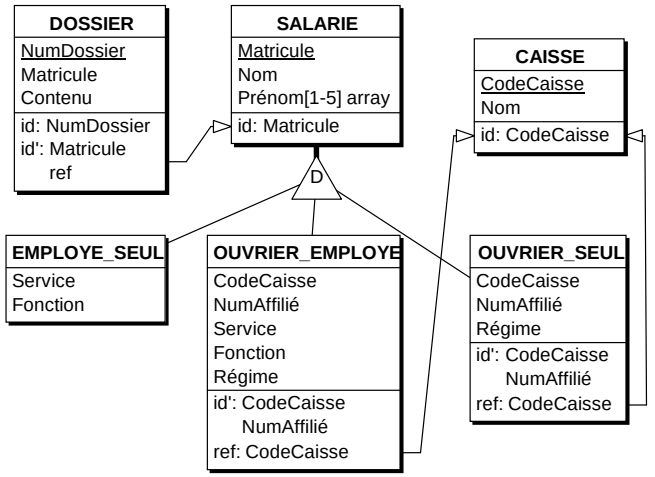
Figure 1.8 - Le monde du travail

Solution

Le principal problème de représentation est celui de la hiérarchie *is-a* sans contrainte de disjonction. On adopte dans un premier temps la transformation de disjonction asymétrique.



Par une transformation symétrique, on obtiendrait le schéma ci-dessous, dans lequel il conviendrait d'ajouter une contrainte d'identifiant secondaire global pour les tables OUVRIER_EMPLOYE et EMPLOYE_SEUL :



19.2 Transformer la hiérarchies *is-a* de la figure 1.9 de manière à la rendre conforme au modèle relationnel objet.

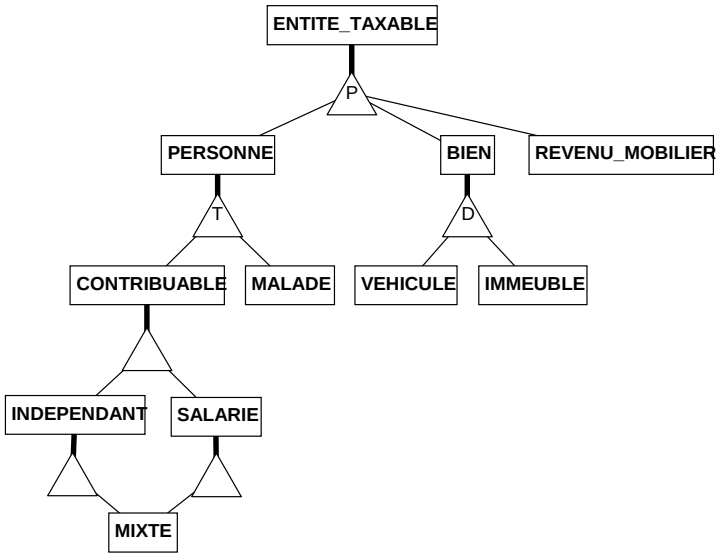
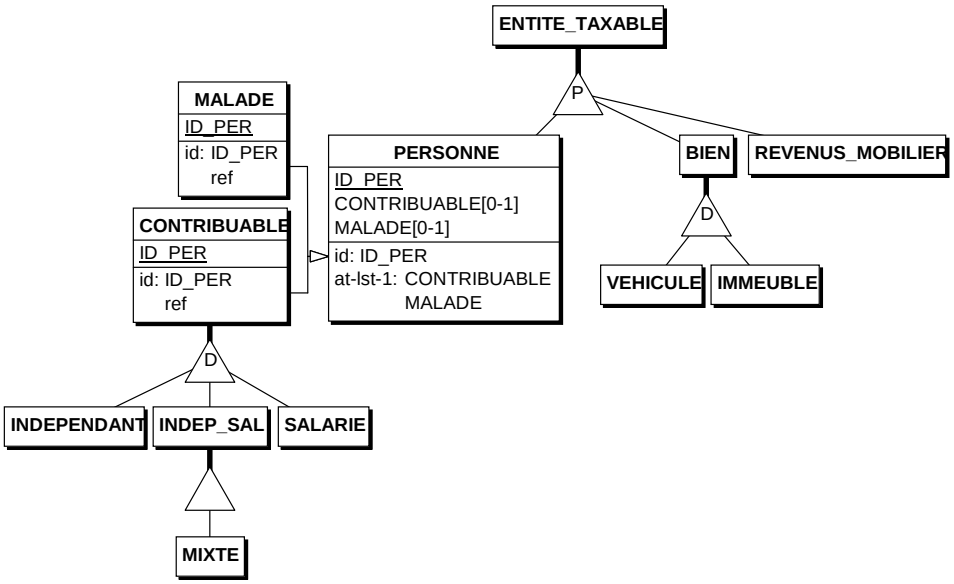


Figure 1.9 - Univers économique

Solution

La sous-hiérarchie (ENTITE_TAXABLE (PERSONNE, BIEN (VEHICULE, IMMEUBLE), REVENU_MOBILIER)) est conforme. La sous-hiérarchie de racine CONTRIBUABLE ne l'est pas, non plus mais est aisée à transformer par disjonction symétrique. Le problème principal est la sous-hiérarchie non disjointe issue de PERSONNE. On la transforme par matérialisation :



19.3 Proposer un schéma relationnel objet pour le schéma conceptuel de la figure 1.10.

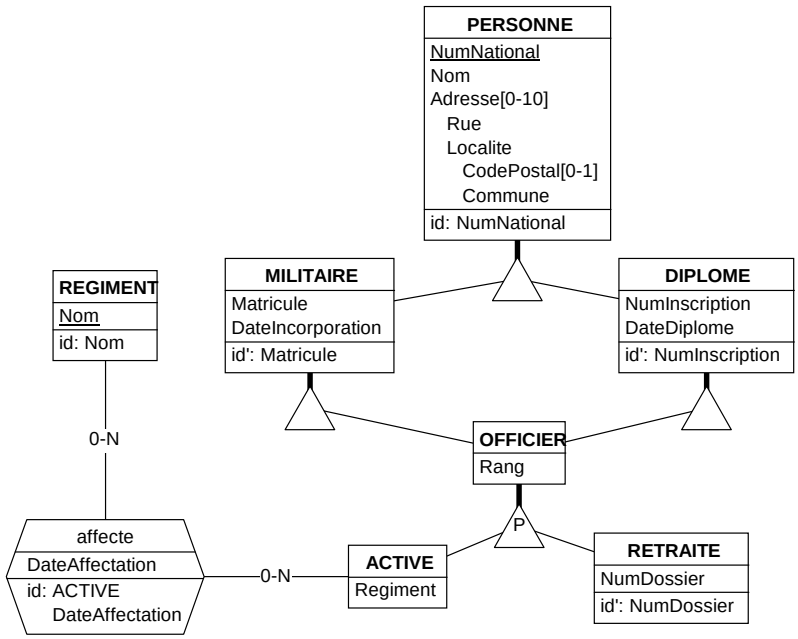
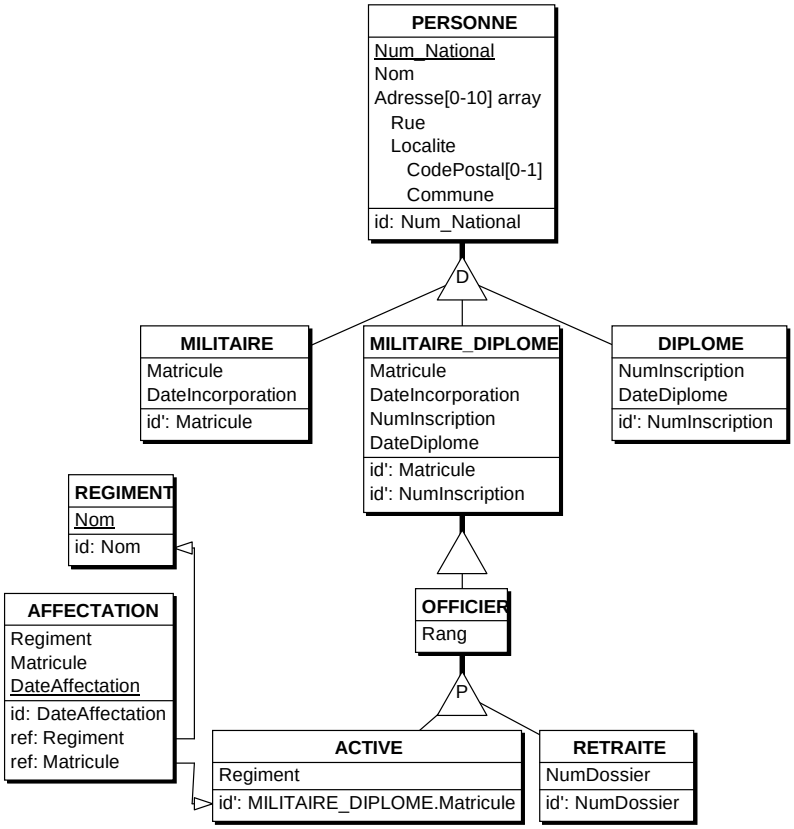


Figure 1.10 - Corps d'armée

Solution

Malgré la hiérarchie sans disjonction et l'héritage multiple, ce schéma se traduit de manière assez naturelle, grâce à l'unique sous-type de MILITAIRE et DIPLOME. On convertit les sous-type de PERSONNE par disjonction symétrique, de manière à obtenir le schéma ci-dessous.



19.4 Transformer le schéma de la figure 1.11 en schéma relationnel objet.

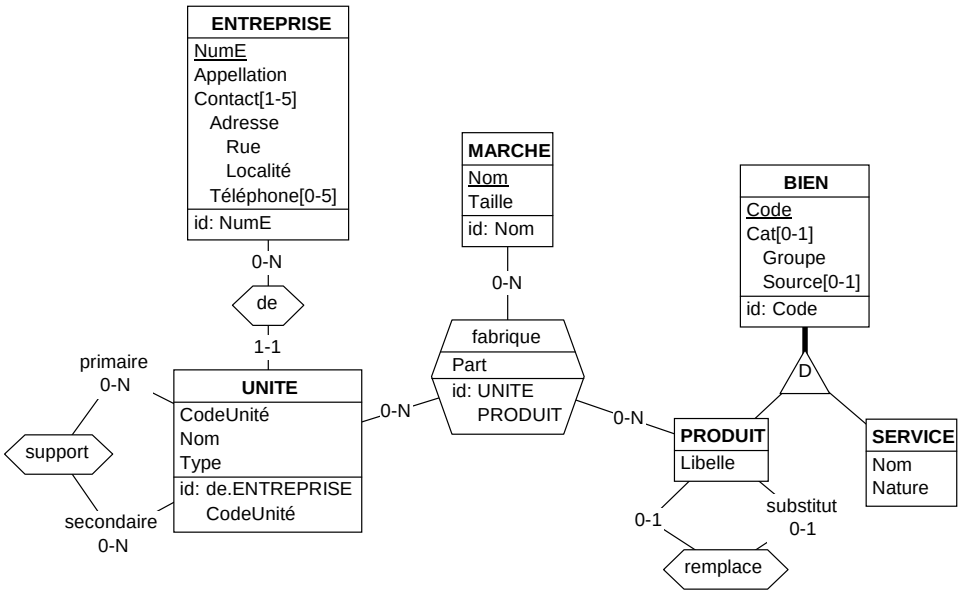


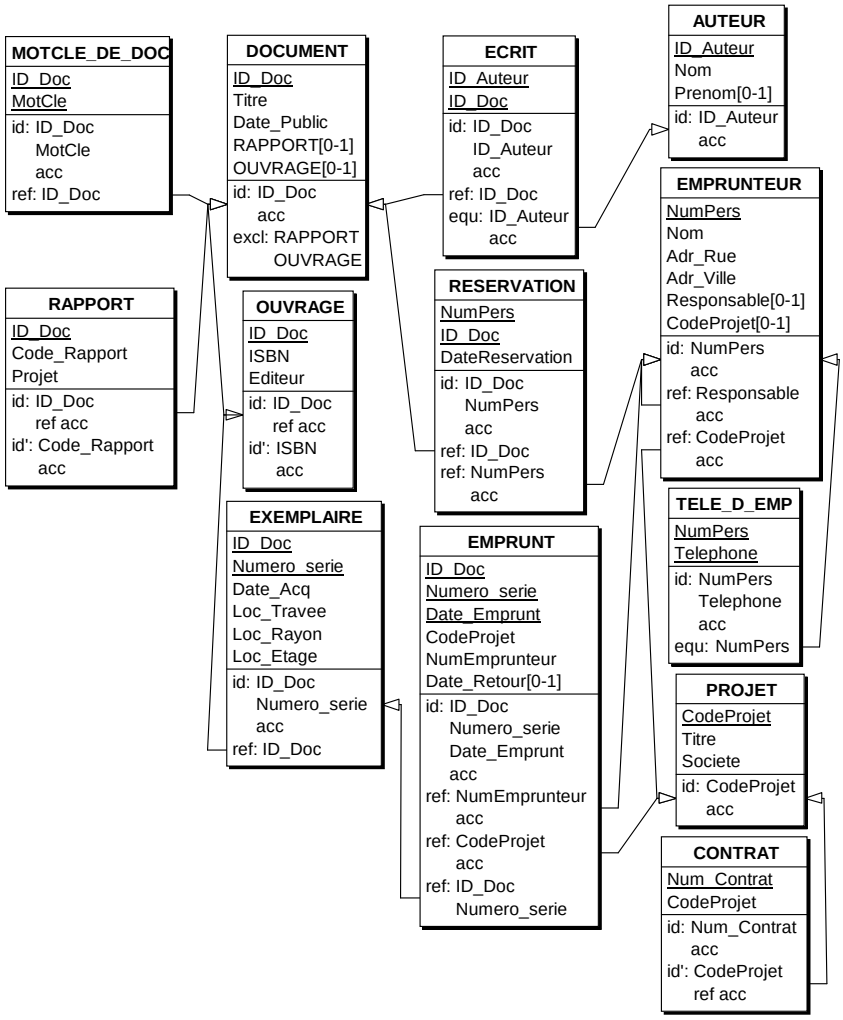
Figure 1.11 - Fabrication

19.5 Développer un plan de transformation relationel objet pour les diagrammes de classes UML.

A.20 CHAPITRE 20 - CONCEPTION PHYSIQUE D'UNE BASE DE DONNÉES RELATIONNELLE

20.1 Compléter le schéma logique de la figure 18.33 en y ajoutant les index induits par les structures logiques.

Solution



20.2 Une table LIVRAISON représentant les livraisons d'articles dans une société de distribution. Elle dispose d'un identifiant composé des colonnes (DATE, REF_CLI). La table comprend actuellement 4.000.000 lignes d'une longueur fixe de 400 octets. Il y a en moyenne 1.000 livraisons par jour. Les colonnes de l'identifiant ont des longueurs respectives de 12 et 28 caractères. La table est stockée dans un espace qui lui est réservé, implanté sur un disque du modèle de référence, décomposé en pages de 4 Ko. On retient les deux requêtes typiques, d et c étant des constantes quelconques :

```
select * from LIVRAISON where DATE = 'd' and REF_CLI =  'c'
select * from LIVRAISON where DATE = 'd'
```

Proposer un jeu d'index favorisant ces deux requêtes. Pour chaque index, proposer une technique d'implémentation. Calculer l'espace occupé par la table et ses index. Calculer le temps moyen d'exécution de ces deux requêtes.

Solution

Les requêtes suggèrent deux index, l'un, I1, sur DATE + REF_CLI et le second, I2, sur DATE. Les paramètres sont **Nr** = 4.000.000, **Lr** = 400, **Nc** = 1.000, **Lk1** = 12 + 28 = 40, **Lk2** = 12 et **Lp** = 4.096.

I2 est composé d'un préfixe de I1, on peut l'ignorer si I1 est implémenté par un index trié : index primaire en séquentiel indexé ou index secondaire avec dictionnaire en séquentiel indexé. On choisi la seconde technique pour I1 et écarte I2.

Par précaution, on vérifie que l'utilisation de l'index lors de l'accès selon DATE est utile. On utilise la feuille de calcul "Scan ou index", qui indique que l'accès séquentiel coûte 92 sec. et l'accès par index 12,3 sec. (hors coût index).

Volume de la table. Avec un taux de remplissage typique de 0,8, on obtient **Nrpp** = 8 et **Np** = 500.000 pages.

Calcul de l'index secondaire. Le dictionnaire de valeurs contient 4.000.000 entrées. Il correspond à un fichier séquentiel indexé dont **Nr** = 4.000.000, **Lr** = 12 + 28 + 4 = 44, **Lk** = 12 + 28 = 40, τ = 0,8 et **Lp** = 4.096. La feuille de calcul "ISAM" nous fournit les résultats suivants : **Npb** = 53.764, **Npi** = 734, **n** = 3. Les niveaux d'index occupent respectivement 1, 10 et 723 pages.

Taille de la table. Table + index occupent 500.000 + 53.764 + 734 = 554.498 pages.

Temps d'accès par I1. Avec un tampon de 1 + 10 + 1 = 12 pages, on compte 2 + 1 = 3 accès aléatoires, soit 36,9 msec. Avec un tampon de 1 + 10 + 723 + 1 = 735 pages, il faut 2 accès aléatoires, soit 24,6 msec.

Temps d'accès par I2. On suppose que **la table est en vrac**. Sachant que **Nrpp** = 74,4 pour le dictionnaire (toujours selon ISAM), les **Nc** = 1.000 entrées de même valeur de DATE occupent 1.000 / 74,4 = 14 pages consécutives du dictionnaire, pour un coût (négligeable) de 14 x 0,184 = 2,6 msec. Viennent s'ajouter les accès aléatoires à 1.000 pages de la table, soit 12,3 sec. Total : 12,3 secondes.

Temps d'accès par I2. On suppose à présent que l'index I1 est **du type *clustering***. Les lignes de même valeur de DATE sont consécutives dans les pages de la table. Elles occupent 1.000 / 8 = 125 pages consécutives. Le coût d'accès est de 125 x 0,184 = 23 msec. Le coût total, index + pages de base, est de 2,6 + 23 = 25,6 msec, soit, théoriquement, 12,3 / 0,023 = 535 fois plus rapide.

A.21 CHAPITRE 21 - PRODUCTION DU CODE D'UNE BASE DE DONNÉES

21.1 Traduire le schéma conceptuel correspondant à l'exercice 12.4 en structure de tables. Ecrire le déclencheur SQL qui vérifie la contrainte sur le nombre maximum de participants à chaque session en recherchant les anomalies. Ecrire une requête SQL qui rassemble les données à inclure dans les factures du mois.

21.2 Traduire en SQL chacun des trois schémas qui clôturent la section 3.8.5.

Solution

On ignore les tables R1 et R2, communes aux trois schémas. Les tables sont renommées I (pour inscription) et A (pour attribution).

La peste (3FN). Un déclencheur vérifie qu'en cas d'insertion et de modification, il n'existe pas dans la table I de valeur de PROF qui serait associée *à une autre valeur* de MAT que celle qui est présente dans la ligne à insérer.

```
create table I(ETUD char(20) not null,
               MAT  char(20) not null,
               PROF char(20 not null,
               primary key (ETUD, MAT));

create trigger C_DF_PROF_MAT
before insert or update on I
declare N number
for each row
begin
  select count(*) into :N from I
  where  PROF = new.PROF and MAT <> new.MAT;
  if N > 0 then abort(); end if;
end;
```

Le choléra (FNBC). Le travail du déclencheur est ici plus complexe. Il recherche les inscriptions existantes de l'étudiant (jointure de I et A) pour la même matière que celle du professeur de l'inscription courante. Il ne doit pas en exister. On veillera également à contrôler les autres opérations de mise à jour des données.

```
create table A(PROF  char(20) not null primary key,
               MAT char(20 not null);
create table I(ETUD char(20) not null,
               PROF char(20 not null,
               primary key (ETUD, PROF),
               foreign key (PROF) references R3);

create trigger C_DF_ETUD_MAT_PROF
begin
before insert or update on I
```

```
declare N number
for each row
  select count(*) into N: from I, A
  where  I.ETUD = new.ETUD and I.PROF = A.PROF
  and    MAT in (select MAT from A
                 where  PROF = new.PROF);
  if N > 0 then abort(); end if;
end;
```

La peste et le choléra (FNCE). Le codage est très simple et ne nécessite pas de déclencheur.

```
create table A(PROF  char(20) not null,
               MAT char(20 not null,
               primary key (PROF, MAT),
               unique (PROF);
create table I(ETUD char(20) not null,
               MAT  char(20) not null,
               PROF char(20 not null,
               primary key (ETUD, MAT),
               foreign key (PROF, MAT) references A);
```

21.3 Le code suivant, réputé avoir été généré par l'atelier Rational Rose, a été présenté il y a quelques années dans un ouvrage sur UML comme la traduction d'une structure formée des classes A, B et C, cette dernière héritant de A et de B. Que penser de cette traduction ?

```
create table A(A_Id number(5), primary key (A_Id));
create table B(B_Id number(5), primary key (B_Id));
create table C(A_Id number(5) references A, B_Id number(5)
               references B, primary key (A_Id, B_Id));
```

Solution

Ce code comporte plusieurs erreurs graves. D'une part, tous les composants d'un identifiant primaire doivent être obligatoires. D'autre par, chacune des clés étrangère de C est en elle-même un identifiant. Code corrigé :

```
create table A(A_Id number(5) not null,
               primary key (A_Id));
create table B(B_Id number(5) not null,
               primary key (B_Id));
create table C(A_Id number(5) not null references A,
               B_Id number(5) not null references B,
               primary key (A_Id),
               unique (B_Id));
```

D'autre part, si on admet que A et B héritent d'une super-classe commune (AB), hypothèse adoptée dans le présent ouvrage et justifiée par le fait que A et B partagent des instances communes, alors A.A_Id est une clé étrangère vers la table de AB, et de même pour B. Dans ce cas, A_Id = B.Id dans C (voir chapitre 19), d'où le code minimal suivant :

```
create table C(A_Id number(5) not null references A,
               primary key (A_Id),
               foreign key (A_Id) references B);
```

A.22 CHAPITRE 22 - RÉTRO-INGÉNIERIE D'UNE BASE DE DONNÉES

Voir Annexe F.

A.25 ANNEXE B - SQL, LES ENSEMBLES ET LA LOGIQUE

25.1 Un amateur de bières affirme : *j'aime les brunes et les légères*. Un autre rétorque : *moi c'est les brunes et légères que je préfère*.

Reformuler ces deux assertions sous forme logique à partir des prédicats suivants :

- j'aime(X) : j'aime la bière X;
- brune(X) : la bière X est brune;
- légère(X) : la bière X est légère.

25.2 Formuler en français la négation de chacune des affirmations suivantes :

- tous les hommes sont mortels
- s'il pleut, la route est mouillée
- si tu ne manges pas tes épinards, tu n'auras pas de dessert
- tout administrateur est responsable pénalement
- il existe des informaticiens qui n'entendent rien à la logique
- toute commande comprend au moins un détail.

25.3 On tient pour vraie la proposition suivante : *s'il ne pleut pas, je n'emporte pas mon parapluie*. Quelles sont, parmi les propositions suivantes, celles qui sont également vraies ?

- si j'emporte mon parapluie, c'est qu'il pleut
- si je n'emporte pas mon parapluie, c'est qu'il ne pleut pas
- j'emmène mon parapluie seulement s'il pleut
- je n'emmène pas mon parapluie seulement s'il ne pleut pas
- il ne pleut pas seulement si je n'emporte pas mon parapluie
- il pleut seulement si j'emporte mon parapluie
- il arrive qu'il pleuve et que je n'emporte pas mon parapluie
- il arrive qu'il pleuve et que j'emporte mon parapluie
- il arrive qu'il ne pleuve pas et que je n'emporte pas mon parapluie
- il arrive qu'il ne pleuve pas et que j'emporte mon parapluie.

25.4 Ambiguïté du langage naturel ! On considère les vers suivants (P. Perret, *L'oiseau dans l'allée*) :

Papa Ronsard qu'était pas une cloche à fromage
Disait qu'il faut danser quand la musique joue

Soient les deux propositions D = "*on danse*" et M = "*la musique joue*". Comment interpréter le second vers :

- $D \Rightarrow M$.
- $M \Rightarrow D$.
- $D \Leftrightarrow M$.

Exprimer les requêtes ci-dessous en SQL.

25.5 Rechercher les clients qui, s'ils ont un compte négatif, ont passé au moins une commande. Quelle est la différence avec la requête suivante : *rechercher les clients qui ont un compte négatif et ont passé au moins une commande* ?

25.6 On considère comme ci-dessus les clients qui, s'ils ont un compte négatif, ont passé au moins une commande. Rechercher les autres clients.

25.7 Rechercher les commandes qui, si elles référencent des produits en sapin, référencent aussi des pointes en acier.

25.8 Rechercher les clients qui ont commandé le produit PA60 ou le produit PA45, mais pas les deux.

25.9 Rechercher les clients qui, s'ils ont commandé le produit PA60, en ont commandé plus de 500 unités au total.

A.26 ANNEXE C - APPLICATIONS AVANCÉES DES BASES DE DONNÉES

C.1 Les structures ordonnées

26.1 Construire une table qui à chaque numéro de client, dont le compte est X, associe le numéro d'un autre client dont le compte Y est égal ou suit directement X.

Suggestion. On construit la *relation d'ordre partiel* selon COMPTE.

26.2 Afficher les clients par couples, tels que le premier a un compte inférieur ou égal au second.

Suggestion. Il s'agit de la fermeture transitive de la relation d'ordre construite à la question 26.1. On peut la construire de manière très simple.

26.3 Afficher les couples de PRODUITS consécutifs selon l'ordre croissant des valeurs de QSTOCK*PRIX.

26.4 Une station météorologique fixe enregistre les températures journalières observées à midi. Les observations sont consignées dans une table T(Date, Temp). Pour simplifier, les dates sont exprimées sous forme de nombres entiers consécutifs. Chaque jour fait l'objet d'une observation (pas de données manquantes). On désire tirer de cette table une table synthétique P(Temp, de, a) qui indique les périodes successives durant lesquelles la température est restée constante. La table doit respecter la propriété suivante : les températures de deux périodes successives sont différentes.

Ecrire une requête SQL qui calcule le contenu de la table P à partir de celui de la table T.

26.5 On considère la table BOURSE de la figure 2.1. Rechercher les dates auxquelles le titre a gagné le plus de points par rapport à la veille.

Suggestion. Cette requête serait élémentaire si elle pouvait s'exprimer sur la table des écarts.

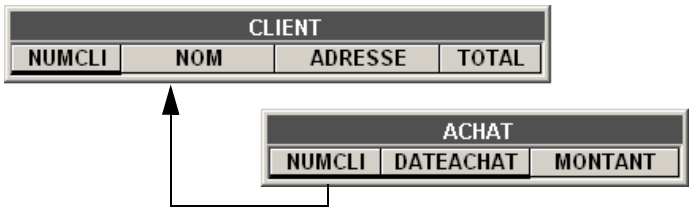
26.6 On complète la table BOURSE par l'ajout d'une colonne TITRE qui représente un titre. De la sorte, cette table représente l'évolution journalière d'un ensemble de titres plutôt que d'un seul. Modifier les requêtes développées jusqu'ici sur la table BOURSE pour qu'elles prennent en compte cette extension.

26.7 Une table RELEVE, de colonnes CLIENT, MOIS et INDICE, d'identifiant (CLIENT, MOIS), reprend pour chaque client et pour chaque mois, le relevé de l'indice de consommation du compteur électrique. Garnir la table

CONSOMMATION, de colonnes CLIENT, MOIS, KWH, de même identifiant que RELEVE, qui reprend la consommation électrique (nombre de kwh = différence entre deux index consécutifs) de chaque client pour chaque mois.

C.2 Les bases de données actives

26.8 On considère le schéma ci-dessous. La colonne TOTAL de chaque CLIENT représente, à tout instant, la somme des MONTANTs des ACHATs du CLIENT. Définir en SQL un schéma qui traduit ces structures et qui gère les valeurs de la colonne MONTANT via un jeu de déclencheurs. On répertoriera d'abord tous les événements susceptibles d'affecter la définition de cette colonne, puis on rédigera pour chacun le déclencheur correctif[8].



26.9 On admet qu'à chaque produit correspond une page HTML qui en décrit les propriétés (exercice 26.38). Rédiger un composant SQL qui rafraîchit automatiquement la page d'un produit dont on vient de modifier une propriété.

26.10 A chaque création d'une ligne d'une table, on désire produire un message en XML qui reprend les données introduites. Proposer une DTD XML adéquate. Rédiger les composants qui créent automatiquement ces messages.

26.11 On ajoute à la table CLIENT une colonne ETAT_CIVIL facultative, dont les valeurs autorisées sont C, M, S, D, V, etc. (codant respectivement les états célibataire, marié, séparé, divorcé, veuf, etc.). Ecrire les composants SQL qui garantissent (1) que cette colonne contient à tout moment une valeur valide ou `null`, (2) que les modifications de valeurs respectent les transitions légales; par exemple, les transitions `null` → C, M → S ou S → D sont valides tandis que C → S ou V → S sont interdites.

Suggestion. La contrainte 1 est statique, et peut être représentée pas des mécanismes élémentaires. La contrainte 2 est dynamique, et réclame la comparaison des valeurs avant et après modification, ce qui n'est possible que dans un déclencheur.

8. Une solution en InterBase est disponible sur le site de l'ouvrage.

26.12 Si le montant d'une commande dépasse 1.000 alors que le compte du client est inférieur à 3.000, la commande est tronquée de manière que son montant ne dépasse pas le seuil de 1.000.

Suggestion. Cette validation peut avoir lieu pour chaque insertion d'un détail. Si le détail viole la règle, son insertion est annulée.

26.13 A tout instant, la table PRODUIT contient une ligne dont la valeur de LIBELLE est 'CADEAU'. Elle représente un article dont une unité est gratuitement ajoutée à toute commande dont le montant dépasse 2.000. Rédigez les composants SQL qui automatisent cette règle.

Suggestion. Le problème est moins simple qu'il n'y paraît, car la réaction est induite par l'insertion des lignes de DETAIL et non par celle de la ligne COMMANDE, ce qui serait prématuré. Si on utilise un déclencheur, celui-ci peut ajouter le *cadeau* dès que le montant dépasse 2.000 *et que le cadeau n'a pas encore été ajouté*.

26.14 On ajoute à la table COMMANDE une colonne MONTANT, qui donne le montant total des détails de chaque commande. Ces valeurs sont évidemment calculables à partir des valeurs de QCOM de DETAIL et de PRIX de PRODUIT. Ecrire les déclencheurs qui gèrent les valeurs de cette nouvelle colonne.

26.15 Nous avons vu dans la section 3.8.5 que la normalisation d'un schéma relationnel dont les **dépendances fonctionnelles** forment un circuit peut conduire à plusieurs solutions dont aucune n'est totalement satisfaisante. Deux d'entre elles (la *peste* et le *choléra*) comportent des contraintes de dépendance fonctionnelle. Supposons que ces schémas soient implémentés en SQL. Proposer un ensemble de déclencheurs qui garantissent la préservation de ces contraintes.

26.16 Une **vue matérialisée** est une *vue* à laquelle est associé un *instantané* créé par une instruction `insert into` *SFW*. Lorsque les données ayant servi à créer l'instantané évoluent, leurs modifications sont propagées vers l'instantané de manière à ce que ce dernier reflète en permanence l'état des tables de base. Imaginez un instantané défini sur les tables de notre exemple et construisez un système de déclencheurs de mise à jour de cet instantané[9].

26.17 On désire ajouter à notre base de données d'exemple un **dispositif de surveillance**. Celui-ci est constitué d'une table de journalisation des opérations de mise à jour et d'un ensemble de déclencheurs qui enregistrent dans cette table tous les événements de modification accompagnés de leur date et heure.

9. Certains SGBD (tel Oracle) prennent en charge ce concept en assurant automatiquement la propagation de ces mises à jour.

26.18 On considère la base de données de la figure 7.2. Un des dangers qui guettent ce type de structure est l'introduction d'un circuit dans le graphe défini par la clé étrangère RESPONSABLE. Par exemple, modifier p2 de manière que son responsable soit p7 introduirait un tel circuit, puisque p2 est responsable indirect de p7. Ecrire un jeu de déclencheurs qui protègent la table PERSONNE contre ces circuits

26.19 La base de données de la figure 7.6 distingue clairement les matières premières des produits finis et semi-finis. En effet, seuls les premiers ont un poids unitaire et un prix unitaire. Le prix unitaire d'un produit dont tous les composants ont un prix unitaire fixé est facile à calculer. Il suffit de propager ces informations des matières premières jusqu'aux produits finis pour compléter la table PRODUIT.

Ecrire un jeu de déclencheurs qui mettent automatiquement à jour le prix unitaire d'un produit fini ou semi-fini chaque fois qu'un composant est ajouté, supprimé ou modifié.

26.20 Considérant notre base de données d'exemple, nous avons toujours admis qu'il ne pouvait exister de détails sans commande, ce que nous avons traduit par la clé étrangère obligatoire DETAIL.NCLI vers CLIENT. Il existe une autre contraintes non exprimée jusqu'ici : *il n'existe pas de commandes sans détail*.

Ecrire un jeu de déclencheurs qui garantisse la validation de cette contrainte. Attention, cette validation doit être soigneusement réfléchie. Elle ne peut en effet jouer pour la création d'une ligne de COMMANDE. On limitera la validation au cas de la suppression et du transfert du dernier détail d'une commande, auquel cas on supprimera celle-ci.

26.21 Ajouter au schéma de la base de données de la figure 2.7 les éléments suivants : des colonnes QREAPPRO, QCOM_MIN et QCOM de la table PRODUIT, indiquant respectivement le seuil et la quantité minimum de rapprovisionnement, et la quantité commandée aux fournisseurs mais non encore livrée; une table FOURNISSEUR; une table OFFRE indiquant à quelles conditions un fournisseur peut livrer un produit : quantité minimum, prix unitaire fixe, frais de transport fixes; enfin une table COM_FOURN qui décrit les commandes de rapprovisionnement effectuées pour un produit et envoyées à un fournisseur.

Lorsqu'un produit tombe en dessous de son seuil de rapprovisionnement, on calcule la quantité à commander, puis on recherche le fournisseur qui peut réapprovisionner au meilleur prix. On passe alors commande à ce fournisseur pour ce produit. Construire un système à base de déclencheur qui crée automatiquement les commandes de rapprovisionnement.

Suggestion. On admet qu'en fin de journée on lance une requête d'ajustement de QSTOCK de PRODUIT (voir 7.10.3). La modification de cette colonne déclenche une réaction de vérification de la quantité à commander éventuellement, de choix du fournisseur et de création d'une commande.

Attention : on ne réagira que si QSTOCK + QCOM est trop faible. On peut d'ailleurs lancer une commande fournisseur alors que d'autres sont encore en attente.

C.3 Les données temporelles

26.22 On considère la table temporelle H_CLIENT de la figure 2.2. Rechercher les numéros des clients qui ne sont plus actifs.

26.23 Toujours dans cette table H_CLIENT, rechercher, pour chaque client, la (les) localité(s) dans laquelle (lesquelles) ce client a été classé dans la catégorie de plus haut niveau dans son existence.

26.24 La table COMMANDE comprend les colonnes (NCLI, DATECOM) qui pourraient être considérées comme formant une clé étrangère temporelle vers la table H_CLIENT. Ecrire une requête qui affiche, pour chaque commande, le nom et la localité du client *au moment indiqué par DATECOM*.

26.25 Afficher pour chaque délégué et chaque localité, le nombre total de jours pendant lesquels il a été responsable de cette localité.

Suggestion. L'intervalle de temps entre deux dates est un entier qui se calcule par la différence entre ces dates.

26.26 Pendant quelles périodes les clients C400 et F011 ont-ils habité dans la même localité ? Indiquer ces localités.

Suggestion. On effectue une auto-jointure temporelle de H_CLIENT sur LOCALITE. Attention aux états jointifs de mêmes valeurs (*coalescing*).

26.27 Donner pour chaque localité les périodes durant lesquelles il y eu au moins un client.

26.28 Quels sont les délégués qui ont eu une interruption de carrière ? Indiquer les périodes d'interruption.

Suggestion. Il y a interruption lorsqu'il existe deux états E1 et E2 successifs qui ne sont pas jointifs (`E1.fin` < `E2.debut`).

26.29 Pour contrôler la gestion de la table H_CLIENT, on propose les déclencheurs suivants. Qu'en pensez-vous ?

Suggestion. En partant d'une table H_CLIENT contenant deux ou trois lignes d'historique, et d'une suite de quelques événements d'évolution des entités représentées, construisez soigneusement la trace de tous les événements et de toutes les opérations définies par ces déclencheurs. A tout instant, observer l'évolution de la table et vérifier si les contraintes d'intégrité sont respectées.

```
create trigger TRG_INSERT_CLI
```

```
before insert on H_CLIENT
for each row
begin
   /* insérer un état courant si l'entité est inconnue
      (= pas encore d'état pour cette entité) */
   if (not exists(select * from H_CLIENT
                  where NCLI= new.NCLI)) then begin
      new.debut = current_timestamp;
      new.fin = 1-01-3000
   end
end

create trigger TRG_B_UPDATE_CLI
before update on H_CLIENT
for each row
begin
   /* définir le nouvel état courant */
   new.debut = current_timestamp;
   new.fin = 1-1-3000;
   /* créer l'état précédent         */
   insert into H_CLIENT values(old.NCLI, old.debut,
   new.debut, old.NOM, old.LOCALITE, old.CAT);
end

create trigger TRG_DELETE_CLI
after delete on H_CLIENT
for each row
begin
   insert into H_CLIENT values (old.NCLI, old.debut,
   current_timestamp, old.NOM, old.LOCALITE, old.CAT);
end
```

26.30 Compléter les déclencheurs de la section 2.4.5 de manière telle qu'ils gèrent la stabilité de l'identifiants d'entité (NCLI), la différence de deux états consécutifs et l'intégrité référentielle (LOCALITE vers H_LOCALITE).

26.31 Il existe d'autres représentations des états historiques d'une collection d'entités. L'une d'elles, bien que peu économique en place occupée, offre une grande simplicité de gestion et d'exploitation. Elle consiste à utiliser deux tables : CLIENT et H_CLIENT. La première contient uniquement les états courants tandis que la seconde contient tous les états, courants et passés. Elles sont définies comme suit :

```
create table CLIENT
(NCLI     char(8)   not null,
 NOM      char(18)  not null,
 LOCALITE char(20)  not null,
 CAT      char(2),
 primary key (NCLI));

create table H_CLIENT
```

```
(NCLI     char(8)   not null,
 debut    timestamp not null,
 fin      timestamp not null,
 NOM      char(18)  not null,
 LOCALITE char(20)  not null,
 CAT      char(2),
 primary key (NCLI,debut))
```

Rédiger les déclencheurs de gestion automatique de l'historique à partir des opérations courantes `insert`, `update` et `delete` effectuées sur la table CLIENT.

26.32 Soit une table EMPLOYE, d'identifiant MATRICULE, et qui donne pour chaque employé, son nom, sa période d'engagement (debut-fin) et le projet auquel il a été affecté. Pour simplifier, les dates sont représentées par des nombres entiers. Si un employé travaille encore aujourd'hui, la date de fin de période est mise à 999. Par exemple, la première ligne indique que l'employé A237, de nom Antoine, travaille sur le projet BIOTECH depuis la date 50. Carlier y a travaillé entre 10 et 40, cette dernière date étant exclue.

EMPLOYE				
MATRICULE	**NOM**	**debut**	**fin**	**PROJET**
A237	Antoine	50	999	BIOTECH
D107	Delecourt	50	999	SURVEYOR
G96	Godin	20	50	SURVEYOR
C45	Carlier	10	40	BIOTECH
N240	Nguyen	30	70	SURVEYOR
A68	Albert	50	999	BIOTECH
M158	Mercier	40	999	SURVEYOR
D122	Declercq	10	999	SURVEYOR

On demande de rédiger un script SQL qui donne, pour chaque projet, l'historique du nombre d'employés.

26.33 La table ci-dessous constitue le journal des installations de logiciels relatives à un parc de machines (on ne montre l'historique que de la machine NEPTUNE). On y enregistre toutes les opérations d'installation et de désinstallation, en indiquant pour chacune la date de l'opération (DATEOPER), l'identifiant de la machine (MACHINE), le nom du logiciel (LOGICIEL) et la nature de l'opération (OPER Œ {"I", "D"}).

OPERATIONS			
DATEOPER	MACHINE	LOGICIEL	OPER
5/01/2006	NEPTUNE	Win XP SP2	I
5/01/2006	NEPTUNE	Win Office XP	I
7/09/2006	NEPTUNE	Adobe PSE 4	I
16/04/2007	NEPTUNE	Open Office	I
16/04/2007	NEPTUNE	Win Office XP	D
19/10/2007	NEPTUNE	Adobe PS CS3	I
21/02/2008	NEPTUNE	Win XP SP2	D
21/02/2008	NEPTUNE	Win VISTA	I
7/04/2008	NEPTUNE	Adobe PSE 4	D

On demande de rédiger une requête SQL qui donne, pour chaque machine, la liste des logiciels disponibles à une date déterminée. On tiendra compte du fait qu'un logiciel installé sur une machine puisse être désinstallé puis réinstallé, éventuellement plusieurs fois.

26.34 On étend la problématique de la question 26.33 en considérant qu'un logiciel puisse faire l'objet ultérieurement d'upgrades, lesquels sont considérés comme des installations. Lors d'une désinstallation d'un logiciel, non seulement celui-ci, mais également tous ses upgrades sont désormais indisponibles. Par exemple, en considérant la séquence d'installations de Win VISTA, Win VISTA SP1, Win VISTA SP2, la désinstallation de Win VISTA entraîne la disparition des upgrades SP1 et SP2 sans qu'il soit nécessaire de mentionner explicitement la désinstallation de ces derniers.

OPERATION(DATEOPER, MACHINE, LOGICIEL, EXTENSION,OPER)

Les opérations mentionnées ci-dessus correspondraient aux lignes suivantes :

```
(21/2/2008, NEPTUNE, Win VISTA, , I)
(13/10/2008, NEPTUNE, Win VISTA, SP1, I)
(4/2/2009, NEPTUNE, Win VISTA, SP2, I)
(6/2/2009, NEPTUNE, Win VISTA, , D)
```

Modifier la requête élaborée pour la question 26.33 pour tenir compte de cette extension.

C.4 La génération de code

26.35 La section 10.5 nous a appris à écrire des requêtes de migration de données. Supposons que la base de données source est toujours celle de la figure 2.7, mais que la base de données cible est d'un format différent. Par exemple, la table DETAIL comporte, outre les colonnes NCOM, NPRO et QCOM, les colonnes NCLI (le numéro du client de la commande), LIBELLE (le libellé du produit commandé) et MONTANT (le montant du détail = QCOM x PRIX). Ecrire la requête qui crée les instructions de migration pour les détails relatifs aux produits en sapin.

26.36 Compléter la procédure de génération d'instructions de création de tables en l'étendant aux autres types de valeurs.

26.37 Compléter la procédure de génération d'instructions de création de tables en y incluant la définition des identifiants primaires et des clés étrangères.

Suggestions. La définition d'un identifiant primaire est elle aussi constituée d'un fragment unique, suivi de fragments multiples (les composants) eux-mêmes suivis d'un fragment unique. Une clé étrangère sera déclarée par un instruction `alter table`. La forme la plus élégante serait sans doute celle d'une procédure SQL (section 9.8) qui encapsule l'ensemble des requêtes de génération et qui permet une gestion algorithmique plus concise et plus lisible des différents cas particuliers (absence d'identifiant par exemple).

26.38 Ecrire un jeu de requêtes qui produise une DTD XML pour l'ensemble des table de la base de données. On ignorera les contraintes d'intégrité telles que les identifiants et les clés étrangères.

26.39 La requête de génération d'instructions de migration développée à la section 2.5.2 (*Migration de données*) fonctionne correctement lorsque la ligne source ne contient pas de valeurs `null`. Modifier cette requête de manière à prendre en compte la présence de valeurs `null`.

Suggestion. Remplacer la valeur extraite de la ligne source par la chaîne '`null`' lorsque cette valeur est `null` (fonction `coalesce` par exemple).

26.40 Modifier la procédure de génération de requêtes de production d'instructions de migration de manière qu'elle produisent des instructions "`insert into`" correctes en présence de valeurs `null` (section 2.5.3, *Génération de migrateurs de données*).

26.41 Modifier la procédure de génération de requêtes de production d'instructions de migration de manière qu'elle produisent les valeurs dans l'ordre de déclaration des colonnes (section 2.5.3, *Génération de migrateurs de données*).

Suggestion. Il manque manifestement une information dans la table des colonne : l'ordre de déclaration des colonnes dans leur table. Appelons CSEQ le numéro de séquence de la colonne. Il faut alors ajouter une colonne correspondante dans la table de travail LIGNE et l'ajouter au critère de tri (`order by`) dans la requête finale de reconstitution des requêtes à partir des fragments. *Remarque* : il faudrait également tenir compte de cet ordre dans la génération des instructions "`create table`".

26.42 Modifier la procédure de génération de requêtes de production d'instructions de migration de manière à éviter la présence d'une virgule derrière la dernière valeur (section 2.5.3, *Génération de migrateurs de données*).

Suggestion. N'insérer une virgule que pour les colonnes dont le numéro d'ordre (CSEQ) n'est pas maximal pour la table.

26.43 Généraliser la requête de génération de requêtes de production d'instructions de migration de manière à traiter toutes les tables de base du schéma.

26.44 A partir des exercices de génération d'instructions de migration et de génération de migrateurs de données, développer des scripts SQL produisant des documents XML.

26.45 Les utilisateurs aimeraient visualiser les commandes au moyen de leur navigateur selon la présentation de la figure 1.

Rédiger la suite de requêtes SFW qui construit automatiquement la page HTML à partir du numéro de la commande.

Suggestions. Cette page étant constituée de fragments uniques et de fragments multiples, on s'inspirera utilement de la procédure de génération des instructions de création de tables. Ici encore on aura intérêt à construire une procédure SQL prenant le numéro de commande en argument[10].

26.46 Poursuivre l'idée de la section 10.5, *Génération de pages HTML*, par la production d'un ensemble de pages décrivant chacune un client et de pages décrivant chacune un produit.

Compléter ensuite la page d'une commande en y ajoutant un lien vers le client et des liens vers chacun des produits référencés par les détails.

Suggestion. Chaque page sera stockée sous un nom qui contient le numéro du client ou du produit. *Exemple* : le client C400 est représenté par la page `CLI_C400.html` et le produit PA60 par la page `PRO_PA60.html`.

10. Une solution complète de cet exercice sous forme de procédures SQL en InterBase est disponible dans [Hainaut, 2001b].

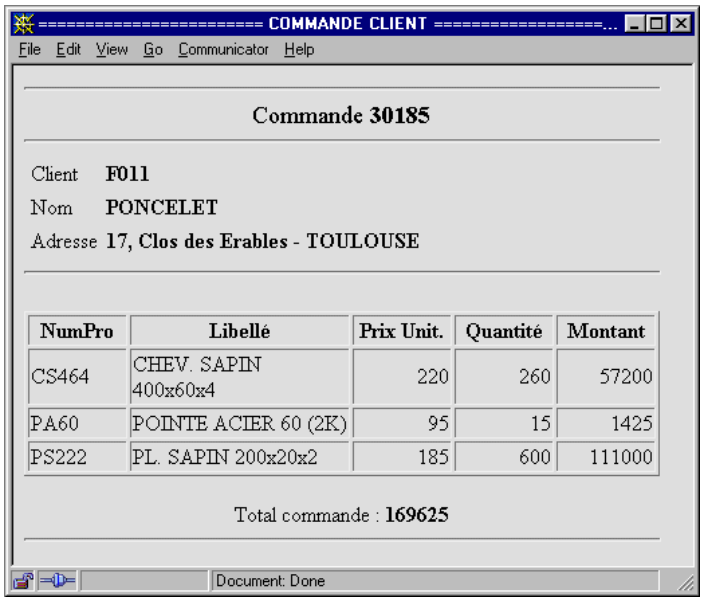


Figure C.1 - Une présentation élégante d'une commande