Date de dernière modification *: 30/6/2015*

Annexe **21**

Production du code d'une base de données

Ce chapitre approfondit les règles de traduction des relations *is-a* et propose deux exercices dont l'un est accompagné de sa solution.

A21.1TRADUCTION DES RELATIONS IS-A

<complément de la section 21.9>

Nous donnons ici le code complet de la traduction du schéma de la figure 21.4 de la section 21.9. La version SQLfast de ce code est aussi disponible dans le fichier **DOCUMENT-is-a.sql**.

A21.1.1 Code des tables de base

```
create table _DOCUMENTE(
     ID_Doc     char(12) not null primary key,
     estRAPPORT integer  not null default 0,
     estOUVRAGE integer  not null default 0,
     Titre      varchar(32) not null,
     unique (ID_Doc,estRAPPORT),
     unique (ID_Doc,estOUVRAGE),
     check(estRAPPORT in (0,1)),
     check(estOUVRAGE in (0,1)),
     check(estRAPPORT + estOUVRAGE <= 1) );

create table _RAPPORT(
     ID_Doc        char(12) not null primary key,
     estRAPPORT    integer  not null default 1,
     Code_Rapport  integer not null,
     Projet        char(20) not null,
     foreign key (ID_Doc,estRAPPORT)
     references _DOCUMENT(ID_Doc,estestRAPPORT)
          on delete cascade on update cascade,
     check(estRAPPORT = 1) );

create table _OUVRAGE(
     ID_Doc        char(12) not null primary key,
     estOUVRAGE    integer  not null default 1,
     ISBN          char(16) not null,
     Editeur       varchar(60) not null,
     foreign key (ID_Doc,estOUVRAGE)
     references _OUVRAGE(ID_Doc,estOUVRAGE)
          on delete cascade on update cascade,
     check(estOUVRAGE = 1) );
```

A21.1.2 Code des vues des sous-types

```
create view RAPPORT(ID_Doc,Titre,Code_Rapport,Projet) as
select D.ID_Doc,D.Titre,R.Code_Rapport,R.Projet
from  _DOCUMENT D,_RAPPORT R
where  D.ID_Doc = R.ID_Doc;

create view OUVRAGE(ID_Doc,Titre,ISBN,Editeur) as
```

```
select D.ID_Doc,D.Titre,O.ISBN,O.Editeur
from  _DOCUMENT D,_OUVRAGE O
where  D.ID_Doc = O.ID_Doc;

create view DOCUMENT_SEUL(ID_Doc,Titre) as
select ID_Doc,Titre
from  _DOCUMENT
where  estRAPPORT = 0 and estOUVRAGE = 0;
```

A21.1.3 Code de la vue du surtype

```
create view DOCUMENT(ID_Doc,estRAPPORT,estOUVRAGE,Titre,
                     Code_Rapport,Projet,
                     ISBN,Editeur) as
select D.ID_Doc,D.estRAPPORT,D.estOUVRAGE,D.Titre,
       R.Code_Rapport,R.Projet,
       O.ISBN,O.Editeur
from  (_DOCUMENT D left outer join _RAPPORT R
         on D.ID_Doc = R.ID_Doc)
       left outer join _OUVRAGE O
         on D.ID_Doc = O.ID_Doc;
```

A21.1.4 Déclencheurs des vues sous-types modifiables

a) Insertion d'un nouveau RAPPORT

```
create trigger TRG_RAPPORT_INSERT
instead of insert on RAPPORT
for each row
declare I number
begin
   select select count(*) in :I from DOCUMENT
   where  ID_DOC = new.ID_DOC;
   if I = 1 then
      abort();
   end if;

   insert into _DOCUMENT(ID_Doc,D.Titre,estRAPPORT)
          values(new.ID_Doc,new.Titre,1);
   insert into _RAPPORT(ID_Doc,Code_Rapport,Projet)
          values(new.ID_Doc,new.Code_Rapport,new.Projet);
end;
```

b) Insertion d'un nouvel OUVRAGE

```
create trigger TRG_OUVRAGE_INSERT
instead of insert on OUVRAGE
for each row
declare I number
begin
```

```
   select select count(*) in :I from DOCUMENT
   where  ID_DOC = new.ID_DOC;
   if I = 1 then
      abort();
   end if;

   insert into _DOCUMENT(ID_Doc,D.Titre,estOUVRAGE)
          values(new.ID_Doc,new.Titre,1);
   insert into _OUVRAGE(ID_Doc,ISBN,Editeur)
          values(new.ID_Doc,new.ISBN,new.Editeur);
end;
```

c) Insertion d'un nouveau DOCUMENT_SEUL

```
create trigger TRG_DOCUMENT_SEUL_INSERT
instead of insert on DOCUMENT_SEUL
for each row
declare I number
begin
   select select count(*) in :I from DOCUMENT
   where  ID_DOC = new.ID_DOC;
   if I = 1 then
      abort();
   end if;

   insert into _DOCUMENT(ID_Doc,D.Titre)
          values(new.ID_Doc,new.Titre,1);
end;
```

d) Suppression d'un RAPPORT

```
create trigger TRG_RAPPORT_DELETE
instead of delete on RAPPORT
for each row
begin
   delete from _DOCUMENT
   where ID_Doc = old.ID_Doc;
end;
```

e) Suppression d'un OUVRAGE

```
create trigger TRG_OUVRAGE_DELETE
instead of delete on OUVRAGE
for each row
begin
   delete from _DOCUMENT
   where ID_Doc = old.ID_Doc;
end;
```

f) Suppression d'un DOCUMENT_SEUL

```
create trigger TRG_DOCUMENT_SEUL_DELETE
instead of delete on DOCUMENT_SEUL
for each row
begin
   delete from _DOCUMENT
   where ID_Doc = old.ID_Doc;
end;
```

g) Modification d'un RAPPORT

```
create trigger TRG_RAPPORT_UPDATE
instead of update on RAPPORT
for each row
begin
   if new.Code_Rapport <> old.Code_Rapport
     update _RAPPORT set Code_Rapport = new.Code_Rapport
     where  ID_Doc = old.ID_Doc;
   end if;
   if new.Projet <> old.Projet
     update _RAPPORT set Projet = new.Projet
     where  ID_Doc = old.ID_Doc;
   end if;
   if new.Titre <> old.Titre
     update _DOCUMENT set Titre = new.Titre
     where  ID_Doc = old.ID_Doc;
   end if;
   if new.Doc_Id <> old.Doc_Id
     update _DOCUMENT set ID_Doc = new.ID_Doc
     where  ID_Doc = old.ID_Doc;
   end if;
end;
```

h) Modification d'un OUVRAGE

```
create trigger TRG_OUVRAGE_UPDATE
instead of update on OUVRAGE
for each row
begin
   if new.ISBN <> old.ISBN
     update _OUVRAGE set ISBN = new.ISBN
     where  ID_Doc = old.ID_Doc;
   end if;
   if new.Editeur <> old.Editeur
     update _OUVRAGE set Editeur = new.Editeur
     where  ID_Doc = old.ID_Doc;
   end if;
   if new.Titre <> old.Titre
     update _DOCUMENT set Titre = new.Titre
     where  ID_Doc = old.ID_Doc;
   end if;
```

```
      if new.Doc_Id <> old.Doc_Id
        update _DOCUMENT set ID_Doc = new.ID_Doc
        where  ID_Doc = old.ID_Doc;
      end if;
   end;
```

i) Modification d'un DOCUMENT_SEUL

```
   create trigger TRG_DOCUMENT_SEUL_UPDATE
   instead of update on DOCUMENT_SEUL
   for each row
   begin
      if new.Titre <> old.Titre
        update _DOCUMENT set Titre = new.Titre
        where  ID_Doc = old.ID_Doc;
      end if;
      if new.Doc_Id <> old.Doc_Id
        update _DOCUMENT set ID_Doc = new.ID_Doc
        where  ID_Doc = old.ID_Doc;
      end if;
   end;
```

A21.1.5 Déclencheurs de la vue surtype modifiable

j) Insertion d'un nouveau DOCUMENT

```
create trigger TRG_DOCUMENT_INSERT
instead of insert on DOCUMENT
for each row
declare I number
begin
   insert into _DOCUMENT(ID_Doc,estRAPPORT,estOUVRAGE,Titre)
   values(new.ID_Doc,new.estRAPPORT,new.estOUVRAGE,new.Titre);

   if new.estRAPPORT = 1
      insert into _RAPPORT(ID_Doc,Code_Rapport,Projet)
      values(new.ID_Doc,new.Code_Rapport,new.Projet);
   end if;

   if new.estOUVRAGE = 1
      insert into _OUVRAGE(ID_Doc,ISBN,Editeur)
      values(new.ID_Doc,new.ISBN,new.Editeur);
   end if;
end;
```

k) Modification d'un DOCUMENT

```
create trigger TRG_DOCUMENT_UPDATE
instead of update on DOCUMENT
```

```
for each row
declare I number
begin
  if new.Titre <> old.Titre
     update _DOCUMENT set Titre = new.Titre
     where  ID_Doc = old.ID_Doc;
  end if;

  if old.estRapport = 1 and new.estRAPPORT = 0
     delete from _RAPPORT
     where ID_Doc = old.ID_Doc;
     update DOCUMENT set estRAPPORT = 0
     where ID_Doc = old.ID_Doc;
  end if;

  if old.estOUVRAGE = 1 and new.estOUVRAGE = 0
     delete from _OUVRAGE
     where ID_Doc = old.ID_Doc;
     update DOCUMENT set estOUVRAGE = 0
     where ID_Doc = old.ID_Doc;
  end if;

  if old.estRapport = 0 and new.estRAPPORT = 1
     insert into _RAPPORT(ID_Doc,Code_Rapport,Projet)
     values(new.ID_Doc,new.Code_Rapport,new.Projet);
     update DOCUMENT set estRAPPORT = 1
     where ID_Doc = old.ID_Doc;
  end if;

  if old.estOUVRAGE = 0 and new.estOUVRAGE = 1
     insert into _OUVRAGE(ID_Doc,ISBN,Editeur)
     values(new.ID_Doc,new.ISBN,new.Editeur);
     update DOCUMENT set estOUVRAGE = 1
     where ID_Doc = old.ID_Doc;
  end if;

  if new.Doc_Id <> old.Doc_Id
     update _DOCUMENT set ID_Doc = new.ID_Doc
     where  ID_Doc = old.ID_Doc;
  end if;
end;
```

l) Suppression d'un DOCUMENT

```
create trigger TRG_DOCUMENT_DELETE
instead of delete on DOCUMENT
for each row
begin
   delete from _DOCUMENT
   where ID_Doc = old.ID_Doc;
end;
```

A21.2TRADUCTION DES RELATIONS IS-A (APPPROCHE CLASSIQUE)

<complément de la section 21.9>

Cette section reprend les matériaux de la deuxième édition de l'ouvrage relatifs à la traduction des relations is-a en SQL2. Elle propose des techniques alternatives de cette traduction.

La traduction correcte d'une structure composée d'une table surtype et de ses tables sous-types n'est pas une tâche triviale. Examinons le schéma de la figure A21.1. Il dérive d'une structure constituée du surtype DOCUMENT et des deux sous-types RAPPORT et OUVRAGE disjoints non couvrants (**D** et ¬**T**).

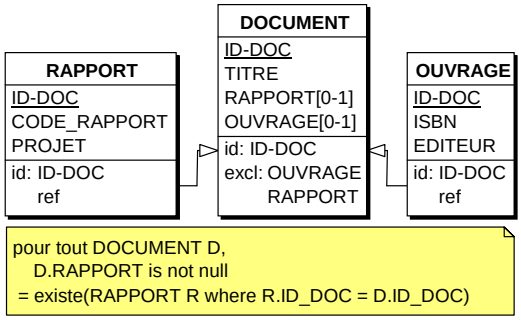

Figure A21.1 - Contrainte de sous-type (pour RAPPORT seulement)

Pour l'utilisateur, l'enregistrement d'un rapport consiste en l'insertion d'une ligne de DOCUMENT suivie de l'insertion d'une ligne de RAPPORT :

```
-- création d'un rapport
insert into DOCUMENT(ID_DOC,TITRE) values('12','Etude finale');
insert into RAPPORT values('12','R0177','BIOGEN');
```

La suppression d'un rapport s'effectuera par la suppression de la ligne de DOCUMENT correspondante.

```
-- suppression d'un rapport
delete from DOCUMENT where ID_DOC='12';
```

Le retrait du statut de *rapport* d'un document consiste à supprimer la ligne de RAPPORT correspondante.

```
-- retirer le statut de rapport à un document
delete from RAPPORT where ID_DOC='12';
```

Pour accorder le statut de *rapport* à un document sans statut on lui associe une ligne de RAPPORT.

```
-- un document sans statut devient un rapport
insert into RAPPORT values('12','R0177','BIOGEN')
```

On implémentera les règles suivantes qui garantissent le respect des contraintes du schéma. On remarque que la contrainte d'exclusion excl n'est pas gérée explicitement. Elle en effet garantie par la gestion des insertions et suppressions de lignes.

- Les colonnes RAPPORT et OUVRAGE sont null à l'insertion d'une ligne de DOCUMENT (via un déclencheur, car une clause default ne suffirait pas).
- Un utilisateur n'est pas autorisé à modifier directement les colonnes RAPPORT et OUVRAGE de DOCUMENT ni les colonnes ID_DOC de RAPPORT et d'OUVRAGE.
- Gestion de la clé étrangère RAPPORT.ID_DOC (OUVRAGE.ID_DOC) du type *cascade*.
- L'insertion d'une ligne de RAPPORT (OUVRAGE) n'est autorisée que si les colonnes RAPPORT et OUVRAGE de la ligne de DOCUMENT correspondante sont null.
- L'insertion d'une ligne de RAPPORT (OUVRAGE) s'accompagne de la mise à "not null" (par exemple `'*'`) de la colonne RAPPORT (OUVRAGE) de la ligne de DOCUMENT correspondante.
- La suppression d'une ligne de RAPPORT (OUVRAGE) s'accompagne de la mise à "null" de la colonne RAPPORT (OUVRAGE) de la ligne de DOCUMENT correspondante.

Le code assigné à la gestion de la table RAPPORT est le suivant :

```
create trigger TRG_DOCUMENT_INSERT_SURTYPE
before insert on DOCUMENT
for each row
begin
  new.RAPPORT = null; new.OUVRAGE = null;
end;

revoke update (RAPPORT,DOCUMENT) on DOCUMENT from public;
revoke update (ID_DOC) on RAPPORT from public;

alter table RAPPORT add constraint FK_RAPPORT_DOC
foreign key (ID_DOC) references DOCUMENT
on delete cascade on update cascade;

create trigger TRG_RAPPORT_INSERT_ISA_DOC
before insert on RAPPORT
for each row
declare I number
begin
  select COUNT(*) in :I from DOCUMENT
  where  ID_DOC = new.ID_DOC
  and   (RAPPORT is not null or OUVRAGE is not null);
  if I = 1 then abort(); end if;
  update DOCUMENT set RAPPORT = '*'
  where  ID_DOC = new.ID_DOC;
end;
```

```
create trigger TRG_RAPPORT_DELETE_ISA_DOC
after delete on RAPPORT
for each row
begin
  update DOCUMENT set RAPPORT = null
  where  ID_DOC = new.ID_DOC;
end;
```

La non-disjonction (¬**D**) modifierait le code de `TRG_RAPPORT_INSERT_ISA_DOC` comme suit :

la condition `(RAPPORT is not null or OUVRAGE is not null)`
se réduit à `(RAPPORT is not null)`.

Attention aux hiérarchies avec contrainte de totalité (T)

En revanche, une contrainte de couverture (**T** ou **P**) va entraîner des modifications substantielles dans la gestion des données, tant pour l'utilisateur que pour le SGBD. En effet, il n'est plus possible désormais d'insérer une ligne de DOCUMENT puis une ligne de RAPPORT, ni l'inverse d'ailleurs. Il est nécessaire d'insérer les deux lignes **simultanément**. Tout comme pour la gestion de la clé étrangère equ (section 21.7), l'idée d'imposer au programmeur de procéder à ces insertions dans une transaction n'est pas recommandable. La définition de procédures ou de méthodes de table typée (avec ou sans *is-a*) en revanche répond à cette contrainte. Quelques exemples :

- RAPPORT_CREER : si elle n'existe pas déjà, insérer une ligne de DOCUMENT; insérer une ligne de RAPPORT ;
- RAPPORT_SUPPRIMER : supprimer la ligne de RAPPORT; si le rapport n'est pas également un ouvrage, supprimer la ligne de DOCUMENT ;
- RAPPORT_MODIFIER : modifier une ou des colonnes de la ligne de DOCUMENT et/ou de la ligne de RAPPORT.

En contrepartie, on interdit la modification directe du contenu des tables DOCUMENT et RAPPORT (`revoke from public`). Cette solution soustrait cependant les concepts d'ouvrage et de document à la puissance de manipulation d'SQL[1]. Il existe deux autres solutions plus satisfaisantes et plus légères, dont l'applicabilité dépend cependant de la puissance du SGBD.

1. Si le SGBD autorise les **mises à jour sur une vue** formée de la jointure de deux tables complètes sur leurs identifiants respectifs, alors on peut effectuer les opérations insert, delete et update sur une telle vue :

```
create view DOC_RAPPORT(ID_DOC,TITRE,CODE_RAPPORT,PROJET)
as
select D.ID_DOC,TITRE,CODE_RAPPORT,PROJET
from   DOCUMENT D, RAPPORT S
where  D.ID_DOC = R.ID_DOC;

-- création d'un rapport
```

1. Les formes `delete from ... where` et `update ... where` sont inutilisables. On retrouve le conflit classique entre l'encapsulation de l'approche OO et la puissance d'expression de SQL.

```
insert into DOC_RAPPORT(ID_DOC,TITRE,CODE_RAPPORT,PROJET)
values ('12','Etude finale','R0177','BIOGEN');

-- suppression d'un rapport
delete from DOC_RAPPORT where ID_DOC='12';

-- modification d'un rapport
update DOC_RAPPORT set TITRE='Etude finale', PROJET='BIOSTAT'
where ID_DOC='12';
```

2. Si le SGBD admet les **triggers instead of** sur une vue[2], alors on crée une vue sur la jointure de DOCUMENT et RAPPORT puis on définit les déclencheurs qui se substituent aux opérations insert, delete et update sur cette vue :

```
create view DOC_RAPPORT(ID_DOC,TITRE,CODE_RAPPORT,PROJET)
as etc.

create trigger TRG_INSERT_DOC_RAPPORT
instead of insert on DOC_RAPPORT
for each row
begin
  insert into DOCUMENT(ID_DOC,TITRE,RAPPORT)
         values(new.ID_DOC,new.TITRE,'*');
  insert into RAPPORT
         values(new.ID_DOC,new.CODE_RAPPORT,new.PROJET);
end;
```

Il est bien évident, *qui peut le plus peut le moins*, que ces deux techniques sont parfaitement applicables aux autres configurations des contraintes de sous-types. La modification directe du contenu de ces tables doit toujours être interdite.

2. C'est le cas d'Oracle, DB2, SQL Server, InterBase, FireBird, SQLite et PostgreSQL (via create rule). Pas encore disponible chez MySQL.

A21.3EXERCICES DU CHAPITRE 21

A21.1 Traduire le schéma conceptuel correspondant à l'exercice 13.4 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.

A21.2 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);
```

A21.3 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 section 18.6.6), 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);
```