Exercices

Les exercices suivants mettent en œuvre les concepts décrits dans le chapitre 1. Il s'agit d'écrire des expressions sous la forme d'opérateurs relationnels interrogeant la base exemple.

EXERCICE 1 PROJECTION ET RESTRICTION

Énoncé

Donnez l'expression permettant d'extraire le mail et le nom des utilisateurs appartenant à l'organisation « IUT Blagnac ».

Solution

$$R1 = \Pi_{(USR\_MAIL,USR\_NOM)}[\sigma_{(USR\_ORGANISATION='IUT\ Blagnac')}[S\_NEWS.UTILISATEUR\_USR]]$$

Seule la table S_NEWS.UTILISATEUR_USR est manipulée. Il faut d'abord restreindre la table aux utilisateurs de l'organisation « IUT Blagnac », puis ne conserver (par projection) que les colonnes USR_MAIL et USR_NOM.

EXERCICE 2 UNION

Énoncé

Donnez l'expression permettant d'extraire le nom de tous les utilisateurs de l'organisation « IUT Blagnac » suivi du nom de tous les forums.

Solution

$$R2 = U[\Pi_{(USR\_NOM)}[\sigma_{(USR\_ORGANISATION='IUT\ Blagnac')}[S\_NEWS.UTILISATEUR\_USR]], \Pi_{(FRM\_NOM)}[S\_NEWS.FORUM\_FRM]]$$

L'union de la colonne USR_NOM la table S_NEWS.UTILISATEUR_USR restreinte aux utilisateurs de l'organisation « IUT Blagnac », avec la colonne FRM_NOM la table S_NEWS.FORUM_FRM répond à cette question.

EXERCICE 3 DIFFÉRENCE

Énoncé

Donnez l'expression permettant d'extraire les numéros des forums sur lesquels aucun utilisateur ne s'est inscrit.

Solution

$$R3 = -[\Pi_{(FRM\_ID)}[S\_NEWS.FORUM\_FRM], \Pi_{(FRM\_ID)}[TJ\_INSCRIS\_NEWS.ISC]]$$

En soustrayant à l'ensemble de tous les numéros de forums (première projection) ceux pour lesquels un utilisateur s'est inscrit (deuxième projection), on obtient la solution.

EXERCICE 4 INTERSECTION

Énoncé

Donnez l'expression permettant d'extraire les numéros des forums qu'ont les utilisateurs de numéro 1 et 3 en commun.

Solution

$$R4 = \cap\ [\Pi_{(FRM\_ID)}[\sigma_{(USR\_ID=1)}[TJ\_INSCRIS\_NEWS.ISC]],$$
$$\Pi_{(FRM\_ID)}[\sigma_{(USR\_ID=3)}[TJ\_INSCRIS\_NEWS.ISC]]]$$

Il faut composer deux ensembles : le premier est constitué des numéros de forums pour lesquels l'utilisateur de numéro 1 s'est inscrit, le second des numéros de forums pour lesquels l'utilisateur de numéro 3 s'est inscrit. L'intersection de ces deux ensembles permet d'obtenir l'extraction voulue.

EXERCICE 5 JOINTURE

Énoncé

Donnez l'expression permettant d'extraire le nom des utilisateurs ayant déjà visité un forum.

Solution

$$R5 = \Pi_{(USR\_NOM)}[\bowtie_{(USR\_ID=USR\_ID)}[S\_NEWS.UTILISATEUR\_USR, TJ\_INSCRIS\_NEWS.ISC]\ ]$$

Il faut joindre deux ensembles : le premier constitué des utilisateurs (pour extraire le nom) et le second composé par les utilisateurs inscrits à au moins un forum (table `TJ_INSCRIS_NEWS.ISC`). La clause de jointure porte sur le champ commun à ces deux ensembles : le numéro d'utilisateur.

EXERCICE 6 JOINTURE

Énoncé

Donnez l'expression permettant d'extraire le nom et le mail des utilisateurs abonnés à un (ou plusieurs) fournisseur(s) utilisant le serveur d'adresse IP 193.54.227.1.

Solution

$$R6 = \Pi_{(USR\_NOM,USR\_MAIL)}[\bowtie_{(USR\_ID=USR\_ID)}[S\_NEWS.UTILISATEUR\_USR,$$
$$\bowtie_{(FAI\_ID=FAI\_ID)}[S\_NEWS.ABONNER\_ABN,$$
$$\bowtie_{(FAI\_ID=FAI\_ID)}[S\_NEWS.FOURNINES\_NEWS.FAI,$$
$$\sigma_{(SRV\_ADR\_IP='193.54.227.1')}[TJ\_SERVEUR\_SRV]]]]]$$

Il faut d'abord extraire les fournisseurs utilisant le serveur d'adresse IP 193.54.227.1. Cette opération est faite dans la dernière jointure. Ensuite, il faut joindre ce résultat à l'ensemble composé des numéros d'abonnés (table `S_NEWS.ABONNER_ABN`). Cette opération est réalisée dans la deuxième jointure. Une fois les numéros d'utilisateurs obtenus, il suffit de faire une jointure avec la table des utilisateurs pour obtenir leur nom et mail par projection.

EXERCICE 7 DIVISION

Énoncé

Donnez l'expression permettant d'extraire le nom et le mail des utilisateurs inscrits à tous les forums existants.

Solution

$$R7 = \Pi_{(USR\_NOM,USR\_MAIL)}[\bowtie_{(USR\_ID=USR\_ID)} [\div [\Pi_{(USR\_ID,FRM\_ID)}[TJ\_INSCRIS\_NEWS.ISC], \Pi_{(FRM\_ID)}[S\_NEWS.FORUM\_FRM]], S\_NEWS.UTILISATEUR\_USR]]$$

Il faut rendre homogène les ensembles à diviser :

- Les numéros de tous les forums (quotient de la division), opération réalisé par la troisième projection.
- Les numéros d'utilisateurs et les numéros de forums auquels ils sont inscrits, opération réalisée par la deuxième projection (dividende de la division).

La division compare donc le premier ensemble à l'ensemble de référence (tous les forums). Il en résulte les numéros des utilisateurs inscrits à tous les forums. L'extraction de ces numéros permet, par le recours à une jointure avec la table des utilisateurs, d'obtenir leur nom et mail par projection.

Chapitre 2

Exercices

Les exercices qui suivent mettent en œuvre les notions présentées dans chapitre 2.

EXERCICE 1 TYPES DE DONNÉES

Énoncé

Définissez le type de donnée le mieux approprié pour spécifier :

a un nom de jour de la semaine ;

b un nom de mois de l'année ;

c un numéro de semaine ;

d un trigramme ;

e une entrée de mot pour un dictionnaire de traduction ;

f un minutage.

Solution

a

```
CHAR(8)
```

est suffisant car aucun nom de jour n'a plus de 8 caractères. En outre, aucun nom de jour n'étant constitué de caractères diacritiques, il n'est pas nécessaire de préciser une collation. On peut aussi utiliser un VARCHAR(8).

b

```
CHAR(9) COLLATE LATIN1
```

est suffisant car aucun nom de mois n'a plus de 9 caractères. Mais comme certains noms de mois sont constitués de caractères diacritiques (février par exemple), il est recommandé de préciser une collation. On peut aussi utiliser un VARCHAR(9) COLLATE LATIN1.

c

```
SMALLINT
```

paraît suffisant car la numérotation des semaines va de 1 à 53.

d

```
CHAR(3)
```

est parfait car un trigramme est composé de trois lettres non diacritiques. Préciser la collation est donc inutile.

e

```
NVARCHAR(36)
```

semble être une bonne mesure pour une entrée de dictionnaire multilingue.

f

```
FLOAT
```

permet de définir un minutage décimal.

EXERCICE 2 DOMAINES

Énoncé

Définissez le domaine (et ses éventuelles contraintes) le mieux approprié pour spécifier :

a le sexe d'un animal ;

b une adresse machine (MACADDRESS) ;

c un e-mail ;

d un poids ;

e une température en degrés centigrades ;

f un booléen ;

g un nom de ville pour une adresse ;

h un groupe sanguin.

Solution

a

```
CREATE DOMAIN D_SEXE_ANIMAL
AS CHAR(1)
CONSTRAINT CK_D_SEXANIMAL_VALEURS CHECK(VALUE IN ('M', 'F', 'A', 'H'))
```

En effet, le sexe d'un animal peut être mâle, femelle, asexué ou hermaphrodite.

b

```
CREATE DOMAIN D_MACADDRESS
AS VARBINARY(48)
```

En effet, une adresse de machine est composée de 12 octets, soit 48 bits.

c

```
CREATE DOMAIN D_MAIL
AS VARCHAR(127)
CONSTRAINT CK_D_MAIL_CONTAIN_AT CHECK(POSITION('@' IN VALUE) > 0)
```

Un nom d'e-mail ne peut excéder 63 caractères pour chaque partie et doit contenir obligatoirement le caractère arobase (@). La fonction SQL POSITION permet de déterminer la position d'une sous-chaîne dans une chaîne. En principe, les seuls caractères autorisés dans la composition des noms Internet sont les 26 lettres en majuscules ou minuscules et le tiret. Mais ce standard est violé et la spécification RFC 1034 évolue.

d

```
CREATE DOMAIN D_POIDS
AS FLOAT
CONSTRAINT CK_D_POIDS_MINVAL CHECK(VALUE >= 0)
```

Un poids ne saurait être négatif.

e

```
CREATE DOMAIN D_TEMPERATURE_DC
AS FLOAT
CONSTRAINT CK_D_TEMPDC_MINVAL CHECK(VALUE >= -273.0)
```

Une température ne saurait descendre en dessous du « zéro absolu ».

f

```
CREATE DOMAIN D_BOOL
AS BIT(1)
DEFAULT '0'
CONSTRAINT CK_D_BOOL_VAL CHECK (VALUE IS NOT NULL)
```

On peut aussi partir d'un type BOOLEAN :

```
CREATE DOMAIN D_BOOLEAN
AS BOLEAN
DEFAULT FALSE
CONSTRAINT CK_D_BOOL_VAL CHECK (VALUE IS NOT NULL)
```

g

```
CREATE DOMAIN D_NOM_VILLE
AS VARCHAR(32)
CONSTRAINT CK_D_NOM_NOSPACE CHECK (SUBSTRING(VALUE FROM 1 FOR 1) <> ' '),
CONSTRAINT CK_D_NOM_NOSPACE CHECK (VALUE = UPPER(VALUE))
```

Un nom ne doit pas commencer par un espace et doit être en principe spécifié en majuscules. La poste impose que la ligne d'adresse ne dépasse pas 38 caractères, mais le nom de ville doit être précédé du code postal. Un espace doit figurer entre les deux. Il reste donc 32 caractères pour spécifier une ville dans le cadre d'une adresse. On pourrait affiner la validation en interdisant tout caractère autre que les lettres (y compris diacritiques), ainsi que le point, le tiret et l'apostrophe.

h

```
CREATE DOMAIN D_GROUPE_SANGUIN
AS VARCHAR(3)
CONSTRAINT CK_D_GRPSANG_PHENOVAL
   CHECK (SUBSTRING(VALUE FROM 1 FOR 1) IN ('A', 'B', 'O')
      OR  SUBSTRING(VALUE FROM 1 FOR 2) = 'AB'),
CONSTRAINT CK_D_GRPSANG_RHESUSVAL
   CHECK (SUBSTRING(VALUE FROM CHARACTER_LENGTH(VALUE) FOR 1) IN ('+', '-'))
```

Les groupes sanguins sont définis par quatre phénotypes (A, B, AB, O) combinés à deux rhésus (positif noté + et négatif noté –). La fonction CHARACTER_LENGTH renvoie le nombre de caractères d'un littéral. Si nous avions codifié ce domaine en utilisant un CHAR(3), il aurait fallu modifier la seconde contrainte de sorte qu'elle ne comptabilise pas les blancs de complément à l'aide de la fonction TRIM :

```
CREATE DOMAIN D_GROUPE_SANGUIN
AS CHAR(3)
CONSTRAINT CK_D_GRPSANG_PHENOVAL
   CHECK (SUBSTRING(VALUE FROM 1 FOR 1) IN ('A', 'B', 'O')
      OR  SUBSTRING(VALUE FROM 1 FOR 2) = 'AB'),
CONSTRAINT CK_D_GRPSANG_RHESUSVAL
   CHECK (SUBSTRING(VALUE FROM CHARACTER_LENGTH(TRIM(VALUE)) FOR 1) IN ('+', '-'))
```

Une solution plus « brute » consiste à énumérer toutes les valeurs des groupes sanguins dans la contrainte de validation :

```
CREATE DOMAIN D_GROUPE_SANGUIN
AS CHAR(3)
CONSTRAINT CK_D_GRPSANG
   CHECK (VALUE IN ('A+ ', 'A- ', 'B+ ', 'B- ',
                    'O+ ', 'O- ', 'AB+', 'AB-')
```

EXERCICE 3 SYNTAXE DE TYPE

Énoncé

Quelles sont les définitions de type syntaxiquement incorrectes ?

a CHAR(16, 2)

b CLOB (16, 2)

c VARCHAR(16)

Énoncé *(suite)*

d NATIONAL CHAR
e VARBINARY
f FLOAT
g DECIMAL
h DECIMAL(16,2)
i DATE (10)
j TIMESTAMP WITH TIME ZONE '-2'
k TIME (3)
l INTERVAL YEAR(100) TO SECOND
m TIMESTAMP (2)

Solution

Sont incorrects :

a Un seul paramètre doit spécifier la longueur.
b Le deuxième paramètre (non obligatoire) ne peut qu'être K, M ou G.
d Il manque la spécification de longueur.
e Il manque la spécification de longueur.
i Une date ne contient pas de paramètre de spécification de taille.
j On ne précise pas la valeur du décalage horaire dans la définition de type d'un TIMESTAMP WITH TIME ZONE, mais dans l'assignation de sa valeur.
l Un intervalle ne peut pas contenir de mois comme granularité intermédiaire entre les bornes.

EXERCICE 4 SYNTAXE DE VALEURS

Énoncé

Quelles sont les spécifications de valeurs syntaxiquement incorrectes ?

a X'AF16'
b B'AF16'
c DATE '18/11/2004'
d B'010011'
e DATE '2002-11-18'
f TIME '11:16'
g TIMESTAMP '2002-11-18 11:16:22 + 01:00'
h DATE '2004-11-18 20:20:30 000'
i INTERVAL '11:16' HOUR TO MINUTE

Énoncé *(suite)*

j 123,55

k 1E-10

l 3.145 E-1

m '456,78'

Solution

Sont incorrects :

b Seuls des 0 et des 1 peuvent être spécifiés dans un binaire exprimé sous forme binaire.

c Une date doit être spécifiée au format ISO 'AAAA-MM-JJ'.

h Une date ne peut pas contenir de spécification horaire.

j Le séparateur d'un réel ou d'un décimal est le point.

l Il ne doit pas y avoir d'espace dans la définition d'un réel.

EXERCICE 5 CRITIQUE DE TYPE

Énoncé

Critiquez les choix de types suivants :

a DECIMAL(5) pour définir un code postal français.

b CHAR(13) pour définir un code EAN 13.

c DECIMAL(11) pour définir un numéro de compte bancaire.

d CHAR(8) pour un numéro ISSN.

Solution

a Mieux vaut un CHAR(5) avec une contrainte. Sinon le tri sur le code postal remonterait 09800 SAINT JEAN DU CASTILLONNAIS en fin de liste : le zéro en tête n'est ni significatif ni stocké en tant que nombre.

b Un DECIMAL (13) est possible, car ce code comprend un contrôle de cohérence interne qui nécessite des calculs complexes. Il faudra cependant penser à le transtyper à l'affichage ainsi que pour le tri en complétant à gauche avec des zéros si besoin est.

c Un numéro de compte bancaire peut contenir des lettres. Il faut donc un CHAR(11).

d L'ISSN contient en final une lettre de contrôle. Ce type est bien adapté.

EXERCICE 6 SPÉCIFICATION DE DOMAINE

Énoncé

Spécifiez un domaine permettant de stocker les dimensions d'un objet à l'aide d'un tableau.

Solution

```
CREATE DOMAIN D_DIMENSIONS
AS FLOAT ARRAY(3)
CONSTRAINT CK_D_DIM_PRECEDENCE
   CHECK (COALESCE(VALUE[1], 0) >= COALESCE(VALUE[2], 0)
      AND COALESCE(VALUE[2], 0) >= COALESCE(VALUE[3], 0))
```

Il n'est pas idiot de prévoir que l'assignation des dimensions doit commencer par la plus forte valeur pour se terminer par la plus petite. Cela permettra de faire des comparaisons de dimensions plus aisées.

La fonction SQL COALESCE permet de remplacer un marqueur NULL par une valeur spécifique.

EXERCICE 7 CRITIQUE DE DOMAINE

Énoncé

Critiquez ces deux définitions pour un type d'adresse IP :

a

```
CREATE DOMAIN D_IP_PART1
AS CHAR(4)
DEFAULT '000.'
CONSTRAINT CK_IP_PART1_NUMBER CHECK (CAST(SUBSTRING(VALUE FROM 1 FOR 3) AS INTEGER)
BETWEEN 0 AND 255),
CONSTRAINT CK_IP_PART1_POINT CHECK (SUBSTRING(VALUE FROM 4 FOR 1) = '.')
```

adresse IP :

```
D_IP_PART1 ARRAY[4]
```

b

```
CREATE DOMAINE D_IP_PART2
AS CHAR(3)
DEFAULT '000'
CONSTRAINT CK_IP_PART2_NUMBER CHECK (CAST(VALUE AS INTEGER) BETWEEN 0 AND 255)
```

adresse IP :

```
D_IP_PART2 ARRAY[4]
```

Solution

a D_IP_ADRESS1 oblige à spécifier des valeurs avec un point terminal comme '192.168.0.2.'.

b D_IP_PART2 ne spécifie pas le point de séparation des différents membres de l'adresse IP.

En outre, les deux définitions de domaines n'empêchent pas l'absence de spécification de valeur. En l'occurrence, il aurait fallu ajouter une contrainte CHECK (VALUE IS NOT NULL).

EXERCICE 8 CRITIQUE DE DOMAINE

Énoncé

Critiquez cette définition de domaine :

```
CREATE DOMAIN D_DATE_DU_JOUR
AS CHAR(10)
DEFAULT CURRENS_NEWS.DATE
```

Solution Elle utilise une fonction non déterministe, en principe non recommandée.

EXERCICE 9 RELATIONNEL OBJET

Énoncé

Construisez un type table à partir d'un type abstrait permettant de modéliser une personne avec sa date de naissance, différentes adresses, différents e-mails et différents téléphones tous typés. Spécifiez une méthode de calcul de l'âge.

Solution Nous commençons par définir les types devant recevoir les adresses, e-mails et téléphones :

```
CREATE TYPE ADS_NEWS.ADRESSE
AS
    (ADR_ADRESSE      VARCHAR(38) ARRAY[4],
     ADR_CODE_POSTAL  CHAR(8),
     ADR_VILLE        CHAR(32),
     ADR_TYPE         CHAR(16))
NOT INSTANTIABLE
NOT FINAL;

CREATE TYPE ADS_NEWS.EMAIL
AS
    (EML_MAIL    VARCHAR(128),
     EML_TYPE    CHAR(16))
NOT INSTANTIABLE

NOT FINAL;
CREATE TYPE ADS_NEWS.TELEPHONE
AS
    (TEL_NUM     VARCHAR(20),
     TEL_TYPE    CHAR(16))
NOT INSTANTIABLE
NOT FINAL;
```

Nous avons choisi de les rendre NON INSTANTIABLE car ils contribuent à l'élaboration du type ADS_NEWS.PERSONNE. Nous les rendons NOT FINAL dans la mesure où nous nous octroyons le droit de les dériver. Par exemple, le type adresse pourrait être dérivé en type « adresse internationale » en y ajoutant une référence de pays.

```
CREATE TYPE ADS_NEWS.PERSONNE
AS
   (PRS_NOM            CHAR(25),
    PRS_PRENOM         VARCHAR(16),
    PRS_DATE_NAISSANCE DATE,
    PRS_ADRESSES       ADS_NEWS.ADRESSE ARRAY[],
    PRS_EMAILS         ADS_NEWS.EMAIL ARRAY[],
    PRS_TELEPHONES     ADS_NEWS.TELEPHONE ARRAY[],
    METHOD PRS_AGE()   RETURNS FLOAT)
INSTANTIABLE
NOT FINAL;
```

Nous définissons le type ADS_NEWS.PERSONNE comme INSTANTIABLE car il se peut qu'on l'utilise pour y placer des informations relatives à des personnes « génériques ». Il doit être aussi NOT FINAL de sorte qu'on puisse le décliner en employé, client, prospect. Pour prévoir de multiples adresses, e-mails et téléphones, il faut passer par des tableaux.

EXERCICE 10 RELATIONNEL OBJET

Énoncé

À partir de ce que vous avez réalisé à l'exercice 9, spécialisez cette table par héritage en une table d'employés en ajoutant le matricule et la date d'embauche.

Solution

Nous partons du type ADS_NEWS.PERSONNE pour créer un type ADS_NEWS.EMPLOYE INSTANTIABLE et FINAL en y ajoutant le matricule et la date d'embauche :

```
CREATE TYPE ADS_NEWS.EMPLOYE UNDER ADS_NEWS.PERSONNE
AS
   (EMP_MATRICULE      CHAR(8),
    EMP_DATE_EMBAUCHE  DATE)
INSTANTIABLE
FINAL
```

La création de la table S_NEWS.EMPLOYE se fait sous le type ADS_NEWS.EMPLOYE auquel on spécifie que la clé est la référence de ligne, et que sa valeur est générée par le système :

```
CREATE TABLE S_NEWS.EMPLOYE_EMP OF ADS_NEWS.PERSONNE
( REF IS EMP_ID SYSTEM GENERATED )
```

EXERCICE 11 RELATIONNEL OBJET

Énoncé

À partir de ce que vous avez réalisé à l'exercice 9, créez une adresse internationale en référençant une table de pays. Créez une table citoyen avec le domicile, le pays, le lieu de naissance et un mécanisme externe pour stocker les photos d'identité.

Solution

Pour créer la référence de pays, utilisez un type UDS_NEWS.PAY et associez-le à une table de référence :

```
CREATE TYPE UDS_NEWS.PAY
(PAY_CODE     CHAR(3) NOT NULL ,
 PAY_LIBELLE  VARCHAR(64) NOT NULL)

CREATE TABLE TR_PAYS_PAY
(PAY UDS_NEWS.PAY)
```

Dès lors, la création de la table des adresses internationales reprend le type ADS_NEWS.ADRESSE auquel on ajoute la référence à la table TR_PAYS_PAY :

```
CREATE TYPE ADS_NEWS.ADR_INTER
AS
   (ADI_ADRESSE    ADS_NEWS.ADRESSE,
    ADI_PAY        REF UDS_NEWS.PAY SCOPE TR_PAYS_PAY
                      REFERENCES ARE CHECKED
                      ON DELETE SET NULL)
INSTANTIABLE
FINAL
```

Nous devons définir un type citoyen :

```
CREATE TYPE ADS_NEWS.CITOYEN
AS
   (CTN_NOM            CHAR(25),
    CTN_PRENOM         VARCHAR(16),
    CTN_DATE_NAISSANCE DATE,
    CTN_LIEU_NAISSANCE VARCHAR(64),
    CTN_PAY_NAISSANCE  REF UDS_NEWS.PAY SCOPE TR_PAYS_PAY
                          REFERENCES ARE CHECKED
                          ON DELETE SET NULL
    CTN_ADR_INTER      ADS_NEWS.ADR_INTER,
    CTN_PHOTO_IDENT    DATALINK FILE LINK CONTROL
                          INTEGRITY ALL
                          READ PERMISSION DB
                          WRITE PERMISION BLOCKED
                          RECOVERY YES
                          ON UNLINK RESTORE)
INSTANTIABLE
FINAL
```

Pour créer finalement la table des citoyens :

```
CREATE TABLE S_NEWS.CITOYEN_CTN OF ADS_NEWS.CITOYEN
( REF IS CTN_ID SYSTEM GENERATED )
```

EXERCICE 12 RELATIONNEL OBJET

Énoncé

Implémentez un TYPE SQL RECTANGLE avec une méthode permettant de déterminer si le rectangle est en fait un carré.

Solution

```
CREATE DOMAINE D_MESURE_LONGUEUR
AS FLOAT
CHECK (VALUE >= 0)

CREATE TYPE ADS_NEWS.RECTANGLE
AS
   (REC_LONGUEUR D_MESURE_LONGUEUR,
    REC_LARGEUR  D_MESURE_LONGUEUR,
    METHOD REC_IS_CARRE() RETURNS BOOLEAN)
INSTANTIABLE
FINAL
```

Pour cette solution, nous avons créé un domaine permettant de rendre les mesures positives. Dans le type abstrait on implémente la méthode REC_IS_CARRE qui renvoie un booléen. Un exemple d'encodage de cette méthode peut être :

```
CREATE METHOD REC_IS_CARRE()
   FOR ADS_NEWS.RECTANGLE
RETURNS BOOLEAN
SELF AS RESULT
LANGUAGE SQL
DETERMINISTIC
CONTAINS SQL
RETURN NULL ON NULL INPUT
RETURN CASE
          WHEN REC_LONGUEUR = REC_LARGEUR
             THEN TRUE
          ELSE FALSE
       END
```

EXERCICE 13 RELATIONNEL OBJET

Énoncé

Implémentez l'ensemble des domaines, types et méthodes nécessaires à une base de données cartographique d'un SIG (Système d'information géographique). En particulier, on considérera le point, la ligne et le polygone.

Solution

Un point cartographique possède trois composantes : la latitude, la longitude et l'altitude. La latitude et la longitude s'expriment en degrés, minutes et secondes, mais les SIG la stockent de plus en plus de façon décimale (les degrés sont exprimés en nombre réel avec des chiffres après la virgule). La latitude s'exprime en rapport au nord ou au sud et sa plage varie de 0 à 90°. La longitude s'exprime par rapport au méridien de référence (celui de Greenwich) et sa plage de valeur va de -180 à +180°. La valeur de l'altitude est comprise entre 0 (niveau de la mer) et ne peut être supérieure à celle du sommet de l'Everest. Par précaution, nous prévoyons cependant une marge par rapport à ces deux extrêmes.

Première partie : création des domaines.

```
CREATE DOMAINE D_LATITUDE
AS FLOAT
CHECK (VALUE BETWEEN 0.0 AND 90.0)

CREATE DOMAINE D_LATITUDE_ORIENTATION
AS CHAR(4)
CHECK (VALUE IN ('NORD', 'SUD ')

CREATE DOMAINE D_LONGITUDE
AS FLOAT
CHECK (VALUE BETWEEN -180.0 AND +180.0)

CREATE DOMAINE D_ALTITUDE
AS FLOAT
CHECK (VALUE BETWEEN -1000.0 AND 10000.0)
```

Seconde partie : création des types.

```
CREATE TYPE ADS_NEWS.POINT
AS
   (PNS_NEWS.LATITUDE     D_LATITUDE,
    PNS_NEWS.ORIENTATION  D_LATITUDE_ORIENTATION,
    PNS_NEWS.LONGITUDE    D_LONGITUDE,
    PNS_NEWS.ALTITUDE     D_ALTITUDE)
INSTANTIABLE
NOT FINAL

CREATE TYPE ADS_NEWS.LIGNE
AS
   (LGN_POINT1    ADS_NEWS.POINT,
    LGN_POINT2    ADS_NEWS.POINT,
    METHOD LGN_LONGUEUR() RETURNS FLOAT)
INSTANTIABLE
NOT FINAL
```

```
CREATE TYPE ADS_NEWS.PLOT
AS
   (PLS_NEWS.ORDRE     INTEGER,
    PLS_NEWS.POINT     ADS_NEWS.LIGNE)
NOT INSTANTIABLE
FINAL

CREATE TYPE ADS_NEWS.POLYGONE
AS
   (PLG_PLOTS     ADS_NEWS.PLOT ARRAY[],
   (PLG_PLOTS     ADS_NEWS.PLOT ARRAY[],
    METHOD PLG_SURFACE() RETURNS FLOAT)
INSTANTIABLE
FINAL
```

Nous avons créé un type ADS_NEWS.POINT suffisant pour représenter un point d'un SIG comme celui représentant le commune de BARJOLS en France : 43.5553 Nord ; 6.0081 ; 250,3.

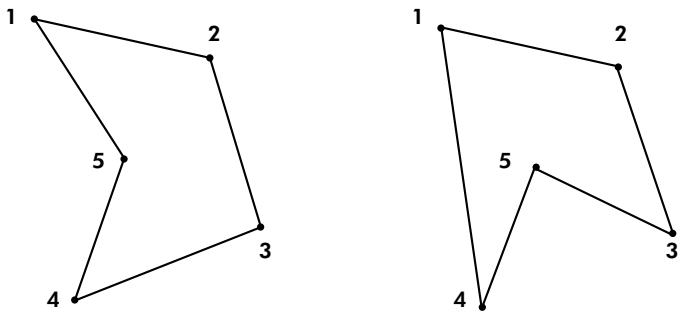
Pour caractériser la ligne, deux points suffisent. Cependant, il faudrait introduire une contrainte dans la construction de la table qui permettrait d'éviter d'avoir des lignes de longueur zéro. Cette contrainte pourrait vérifier que LGN_POINT1 <> LGN_POINT2. Mais on pourrait aussi empêcher des lignes verticales (des falaises en quelques sortes). Dans ce cas, la contrainte deviendrait :

```
NOT (    LGN_POINT1.PNS_NEWS.LATITUDE =    LGN_POINT2.PNS_NEWS.LATITUDE
     AND LGN_POINT1.PNS_NEWS.LONGITUDE =   LGN_POINT2.PNS_NEWS.LONGITUDE
     AND LGN_POINT1.PNS_NEWS.ORIENTATION = LGN_POINT2.PNS_NEWS.ORIENTATION)
```

Nous avons en outre doté le type ligne d'une méthode de calcul de longueur. Pour calculer une telle longueur compte tenu de l'altitude et de la courbure de la terre, il faut utiliser une formule mathématique complexe.

Pour créer le type polygone, nous devons considérer une série de points dont le nombre n'est pas connu. Un tableau ouvert permet cela. Cependant l'ordre des points est une composante essentielle de la définition d'un polygone. Ainsi, avec cinq points, il existe différentes possibilité de tracer un polygone (voir figure 2.2).

Figure 2.2



Exercices

Les exercices qui suivent utilisent les différents objets étudiés dans le chapitre 3. Le premier exercice traite de la convention de nommage. La définition de tables est ensuite abordée ainsi que les contraintes. L'exercice 11 consiste à modifier un schéma existant tandis que les exercices 12 et 13 consistent à en définir un nouveau.

EXERCICE 1 SYNTAXE DES NOMS

Énoncé

Les noms d'objets SQL suivants sont-ils corrects ?

- **a** DEPART
- **b** ARRIVÉE
- **c** DATE
- **d** _WIDE_
- **e** "CREATE"
- **f** #CLIENT
- **g** IBM_db2
- **h** 5e_avenue

Solution

Sont incorrects :

- **b** Le caractère majuscule accentué est interdit.
- **c** DATE est un mot réservé du SQL. Si on veut l'utiliser, il faut l'écrire "DATE" (entre guillemets).
- **d** Un nom d'objet SQL doit commencer par une lettre.
- **f** Le caractère # n'est pas autorisé dans un nom SQL.
- **h** Un nom d'objet SQL ne peut commencer par un chiffre.

EXERCICE 2 SYNTAXE DE TABLE

Énoncé

Expliquez pourquoi la syntaxe de ce CREATE TABLE est incorrecte :

```
CREATE TABLE S_NEWS.MATABLE
( MTB_ID       INTEGER(16) PRIMARY KEY,
  MTB_NOM      CHAR(32) NOT NULL,
  MTB_DATE     DATETIME DEFAULT CURRENT DATETIME,
  MTB_LIBELLE  NATIONAL CHAR VARYING (128),
  MTB_CODE     CHECK (VALUE BETWEEN 'AAAA' AND 'ZZZZ'),
  MTB_ORDRE    INT,
  CONSTRAINT   UNIQUE MTB_DATE, MTB_CODE)
```

Solution

Le type INTEGER ne nécessite aucune spécification de longueur.

Étant donné que la fonction « CURRENT DATETIME » n'existe pas, le type DATETIME est incorrect ; il faut utiliser ici le type TIMESTAMP et la fonction CURRENT_TIMESTAMP.

Il manque la spécification de type à la colonne MTB_CODE.

La contrainte de table UNIQUE doit être nommée et ses composantes (colonnes MTB_DATE et MTB_CODE) doivent être placées entre parenthèses.

EXERCICE 3 SYNTAXE DE TABLE

Énoncé

Expliquez pourquoi la syntaxe de ces ordres SQL est incorrecte :

```
CREATE TABLE S_NEWS.COULEUR_CLR
(CLR_ID       INTEGER,
 CLR_COULEUR  VARCHAR(16),
 CONSTRAINT PK_CLR PRIMARY KEY (CLR_ID))

CREATE TABLE S_NEWS.TYPE_TYP
(TYP_CODE     CHAR(2) NOT NULL PRIMARY KEY,
 TYP LIBELLE  VARCHAR(16),
 CONSTRAINT UK_LIB UNIQUE (TYP_LIBELLE))

CREATE TABLE S_NEWS.VEHICULE_VHC
(VHC_ID               INTEGER NOT NULL PRIMARY KEY,
 VHC_IMMATRICULATION  CHAR(12),
 TYP_CODE             CHAR(2)
 CONSTRAINT PK_TYPE REFERENCES S_NEWS.TYPE_TYP (TYP_CODE),
           ON DELETE SET NULL, ON UPDATE RESTRICT,
 CLR_ID               CHAR(2) REFERENCES S_NEWS.COULEUR_CLR,
 CONSTRAINT VHC_IMMATRICULATION UNIQUE (VHC_IMMATRICULATION))
```

Solution

Table S_NEWS.COULEUR_CLR : la colonne CLR_ID doit être NOT NULL pour participer à la clé primaire.

Table S_NEWS.TYPE_TYP : le nom de la seconde colonne "TYP LIBELLE" contient un blanc.

Table S_NEWS.VEHICULE_VHC : il y a deux virgules en trop dans la définition de la contrainte de référence PK_TYPE. Le type de données de la colonne CLR_ID est incorrect en regard de la table que cette colonne référence. Ce type devrait être INTEGER. Le nom de la contrainte d'unicité VHC_IMMATRICULATION est invalide, car une colonne porte déjà ce nom-là.

EXERCICE 4 CRÉATION DE TABLES

Énoncé

Créez une table pour y stocker les produits à vendre, avec les rubriques suivantes : identifiant, référence magasin, référence fabricant, code EAN13, prix de vente HT. Faites référence aux tables S_NEWS.TAUX_TVA, S_NEWS.RAYON_RYN et S_NEWS.FABRICANS _NEWS.FBQ et mettez en place toutes les contraintes nécessaires.

Solution

```
CREATE TABLE S_NEWS.TAUX_TVA
(TVA_ID    INTEGER NOT NULL PRIMARY KEY,
 TVA_TAUX  FLOAT   NOT NULL CHECK (VALUE > 0.0) )
```

Prévoyons un taux de TVA positif !

```
CREATE TABLE S_NEWS.RAYON_RYN
(RYN_ID       INTEGER  NOT NULL PRIMARY KEY,
 RYN_LIBELLE  CHAR(25) NOT NULL)

CREATE TABLE S_NEWS.FABRICANS_NEWS.FBQ
(FBQ_ID       INTEGER  NOT NULL PRIMARY KEY,
 FBQ_NOM      CHAR(64) NOT NULL)

CREATE TABLE S_NEWS.PRODUIS_NEWS.PRD
(PRD_ID       INTEGER NOT NULL PRIMARY KEY,
 PRD_REF_MAG  CHAR(16) NOT NULL UNIQUE,
 FBQ_ID       INTEGER NOT NULL REFERENCES S_NEWS.FABRICANS_NEWS.FBQ (FBQ_ID),
 PRD_REF_FAB  CHAR(16),
 RYN_ID       INTEGER NOT NULL REFERENCES S_NEWS.RAYON_RYN (RYN_ID),
 PRD_EAN13    CHAR(13),
 PRD_PRIX     FLOAT NOT NULL CHECK(VALUE > 0.0)
 TVA_ID       INTEGER NOT NULL REFERENCES S_NEWS.TAUX_TVA (TVA_ID)
 CONSTRAINT   UK_FBQ_REF UNIQUE (FBQ_ID, PRD_REF_FAB))
```

Mieux vaut que la référence magasin soit unique. Dans le même esprit, la combinaison de la référence du fabricant et de son identifiant doit être unique. Le prix doit être supérieur à zéro.

EXERCICE 5 CRÉATION DE TABLE

Énoncé

À partir des données présentées dans le tableau suivant, proposez une définition pour créer la table S_NEWS.MAINTENANCE_MTN.

Jour	Machine	Numéro	Vitesse	Température	Heure	Événement
Ven	Massicot	147			21:18	Défaut de lame
Sam	Relieuse	63	16		16:15	Arrêt pour maintenance
Jeu	Presse	87	6	62	11:40	Bavure encre
Sam	Relieuse	79	16		17:11	Reprise
Mer	Presse	89	6	55	08:28	Recadrage

Énoncé *(suite)*

Jour	Machine	Numéro	Vitesse	Température	Heure	Événement
Mar	Presse	132	8	68	09:58	Changement encre
Mer	Massicot	111			10:17	Graissage coulisseau

Solution

```
CREATE TABLE S_NEWS.MAINTENANCE_MTN
(MTN_NUMERO       INTEGER NOT NULL PRIMARY KEY,
 MTN_MACHINE      CHAR(16) NOT NULL,
 MTN_JOUR         CHAR(3) NOT NULL
                  CHECK (VALUE IN
                     ('Lun', 'Mar', 'Mer', 'Jeu', 'Ven', 'Sam', 'Dim')),
 MTN_HEURE        TIME NOT NULL,
 MTN_EVENEMENT    VARCHAR(64) NOT NULL,
 MTN_VITESSE      FLOAT CHECK (VALUE > 0),
 MTN_TEMPERATURE  FLOAT CHECK (VALUE > -273)
 CONSTRAINT UK_MAC_JOUR_HEUR UNIQUE (MTN_MACHINE, MTN_JOUR, MTN_HEURE))
```

À l'évidence, la colonne Numéro est unique et peut constituer la clé de la table. Machine, Jour, Heure et Événement sont des colonnes toujours renseignées (NOT NULL). Un CHAR(3) suffit pour la colonne Jour à laquelle on ajoute une contrainte validant la donnée avec les trois premières lettres du jour de la semaine. On ne saurait relever une vitesse négative, pas plus qu'une température inférieure à –273 ; ces deux colonnes font donc l'objet d'une contrainte de validation CHECK. Enfin, il semble indiqué (mais ce n'est pas obligatoire) de définir l'unicité sur les colonnes Machine, Jour et Heure, afin d'éviter de saisir deux événements survenant au même moment pour une même machine.

EXERCICE 6 CRÉATION DE TABLES

Énoncé

À partir des données présentées dans les tableaux suivants, proposez les définitions de tables correspondantes.

COLIS

Numéro	Poids	Longueur	Largeur	Hauteur	Date dépôt
114	7	25	12	7	11/01/2004
178	18	120	78	45	12/01/2004
98	12	87	80	35	10/01/2004
112	3	22	10	8	10/01/2004
127	10	42	42	25	10/01/2004
135	19	45	40	30	10/01/2004
149	1	120	110	100	11/01/2004

Énoncé *(suite)*

EXPÉDITEUR

N°	Sté	Nom	Prenom	Adresse	Ville	CP	Tel
57	Dupont SA	DUPONT	Jean	44 rue François 1er	ORLÉANS	45000	0144789852
63	IBM	WATSON	John	1 rue Rockfeller	MARSEILLE	13000	0423547896
79	Microsoft	GATES	Bill	3 rue Redmond	LYON		
52	Oracle	ELLISON	Harry	6 av. Foch	TOULOUSE	33000	0511224477

DESTINATAIRE

N°	Sté	Nom	Prenom	Adresse	Ville	CP	Tel
52	Oracle	ELLISON	Harry	6 av. Foch	TOULOUSE	33000	0511224477
49	EDF	LEBRUN	Jean	6 rue du Puits	PARIS	75003	0145224822
63	IBM	WATSON	John	1 rue Rockfeller	MARSEILE	13000	0423547896
81	SNCF	PAPIN	Denis	1 rue St Lazare	PARIS	75009	0148757475

TRANSPORT

Expéditeur	Destinataire	Tournée	Jour	Colis
57	63	Marcel	11/01/2004	98
57	63	Marcel	11/01/2004	112
63	81	Raymond	12/01/2004	149
79	52	Raymond	12/01/2004	135
79	52	Gérard	12/01/2004	114
79	52	Raymond	13/01/2004	178
52	49	Marcel	13/01/2004	127

Solution

Le tableau COLIS peut constituer à lui seul une table. Les tableaux EXPÉDITEUR et DESTINATAIRE contiennent certaines informations identiques ; mieux vaut donc les regrouper en une seule table. Le tableau TRANSPORT présente des liens avec les trois autres tableaux. Tout cela peut être modélisé à l'aide des tables suivantes :

```
CREATE TABLE S_NEWS.COLIS_CLS
(CLS_NUMERO      INTEGER NOT NULL PRIMARY KEY,
 CLS_DATE_DEPOT  DATE NOT NULL,
 CLS_POIDS       INTEGER CHECK(VALUE > 0 OR VALUE IS NULL),
 CLS_LONGUEUR    INTEGER CHECK(VALUE > 0 OR VALUE IS NULL),
 CLS_LARGEUR     INTEGER CHECK(VALUE > 0 OR VALUE IS NULL),
 CLS_HAUTEUR     INTEGER CHECK(VALUE > 0 OR VALUE IS NULL))
```

Les dimensions et les poids des colis ne sauraient être négatifs. C'est la raison pour laquelle des contraintes de validation CHECK sont introduites.

```
CREATE TABLE S_NEWS.EXPEDESTIN_XPD
(XPD_NUMERO   INTEGER      NOT NULL PRIMARY KEY,
```

```
 XPD_SOCIETE  VARCHAR(128),
 XPD_NOM      CHAR(25)     NOT NULL CHECK (TRIM(BOTH, VALUE) <> ''),
 XPD_PRENOM   VARCHAR(16),
 XPD_ADRESSE  VARCHAR(128) NOT NULL CHECK (TRIM(BOTH, VALUE) <> ''),
 XPD_VILLE    VARCHAR(32)  NOT NULL CHECK (TRIM(BOTH, VALUE) <> ''),
 XPD_CP       VARCHAR(5)   NOT NULL CHECK (TRIM(BOTH, VALUE) <> ''),
 XPD_TEL      VARCHAR(20),
 XPD_EXPED    BOOLEAN NOT NULL DEFAULT TRUE,
 XPD_DESTIN   BOOLEAN NOT NULL DEFAULT FALSE,
 CONSTRAINT CK_EXPDDEST CHECK (NOT (XPD_EXPED AND XPD_DESTIN)))
```

Nous avons naturellement rendu obligatoires les colonnes contenant les éléments d'adresse. Nous avons en outre placé des contraintes pour empêcher que ces mêmes colonnes restent vides. La fonction TRIM (BOTH supprime les espaces en début et en fin de chaîne. D'autre part, nous avons ajouté deux colonnes :

- XPD_EXPED permet de définir si la ligne correspond à un expéditeur (TRUE) ou non.
- XPD_DESTIN permet de définir si la ligne correspond à un destinataire (TRUE) ou non.

Bien entendu, un même « client » peut être à la fois expéditeur et destinataire, mais il ne peut pas être ni l'un, ni l'autre simultanément. C'est la raison pour laquelle la contrainte de validation CK_EXPDDEST est introduite.

```
CREATE TABLE S_NEWS.TRANSPORS_NEWS.TSP
(CLS_NUMERO     INTEGER NOT NULL PRIMARY KEY,
 XPD_NUMERO_E   INTEGER NOT NULL,
 XPD_NUMERO_D   INTEGER NOT NULL,
 TSP_TOURNEE    VARCHAR(16) NOT NULL CHECK (TRIM(BOTH VALUE) <> ''),
 TSP_JOUR       DATE NOT NULL,
 CONSTRAINT     FK_CLS  FOREIGN KEY (CLS_NUMERO)
                        REFERENCES S_NEWS.COLIS_CLS (CLS_NUMERO),
 CONSTRAINT     FK_XPDE FOREIGN KEY (XPD_NUMERO_E)
                        REFERENCES S_NEWS.EXPEDESTIN_XPD (XPD_NUMERO),
 CONSTRAINT     FK_XPDD FOREIGN KEY (XPD_NUMERO_D)
                        REFERENCES S_NEWS.EXPEDESTIN_XPD (XPD_NUMERO),
 CONSTRAINT     CK_XPD_DEST CHECK (XPD_NUMERO_D <> XPD_NUMERO_E))
```

Le numéro de colis peut parfaitement constituer la clé de cette table. Il servira, de plus, de clé étrangère. La contrainte CK_XPD_DEST vérifie que le destinataire est différent de l'expéditeur (dans le cas contraire, il est probable qu'une erreur de saisie survienne).

On pourrait aussi fondre les tables S_NEWS.COLIS_CLS et S_NEWS.TRANSPORS _NEWS.TSP en une seule. Dans ce cas, il faudrait autoriser les colonnes TSP_TOURNEE et TSP_JOUR à recevoir le paramètre NULL. En effet, au moment de la saisie du colis dans la table, la tournée comme le jour de distribution ne sont sans doute pas encore connus.

Autre solution : créer dans la table S_NEWS.TRANSPORS_NEWS.TSP une clé « artificielle », par exemple un auto-incrément, et considérer la colonne CLS_NUMERO comme une simple clé étrangère. Dans ce cas, il faut ajouter une contrainte d'unicité sur le numéro de colis.

```
CREATE TABLE S_NEWS.TRANSPORS_NEWS.TSP
(TSP_NUMERO     INTEGER NOT NULL PRIMARY KEY
 CLS_NUMERO     INTEGER NOT NULL UNIQUE,
 XPD_NUMERO_E   INTEGER NOT NULL,
 XPD_NUMERO_D   INTEGER NOT NULL,
 TSP_TOURNEE    VARCHAR(16) NOT NULL CHECK (TRIM(BOTH VALUE) <> ''),
 TSP_JOUR       DATE NOT NULL,
 CONSTRAINT     FK_CLS  FOREIGN KEY (CLS_NUMERO)
                        REFERENCES S_NEWS.COLIS_CLS (CLS_NUMERO),
```

```
CONSTRAINT     FK_XPDE FOREIGN KEY (XPD_NUMERO_E)
                       REFERENCES S_NEWS.EXPEDESTIN_XPD (XPD_NUMERO),
CONSTRAINT     FK_XPDD FOREIGN KEY (XPD_NUMERO_D)
                       REFERENCES S_NEWS.EXPEDESTIN_XPD (XPD_NUMERO),
CONSTRAINT     CK_XPD_DEST CHECK (XPD_NUMERO_D <> XPD_NUMERO_E))
```

EXERCICE 7 MODIFICATIONS DE TABLES

Énoncé

En partant de la solution de l'exercice précédent, considérez que les colonnes Tournée et Jour du tableau TRANSPORT sont des tables nommées S_NEWS.TOURNEE_TRN et S_NEWS.PLANNING_PLN. Écrivez les ordres de création de ces tables et modifiez les tables déjà définies lors du précédent exercice pour y introduire les références à ces nouvelles tables. Pensez à minimiser l'espace de stockage des informations.

Solution

On peut créer ces deux tables ainsi :

```
CREATE TABLE S_NEWS.TOURNEE_TRN
(TRN_ID       INTEGER NOT NULL PRIMARY KEY,
 TRN_TOURNEE  VARCHAR(16) NOT NULL UNIQUE)
```

Pour minimiser l'espace de stockage, on utilise une clé de type INTEGER qui ne nécessite que 2 octets, alors que le nom de la tournée est codifié sur 16 octets.

```
CREATE TABLE S_NEWS.PLANNING_PLN
(PLN_DATE   DATE NOT NULL PRIMARY KEY)
```

Il est inutile de créer dans le planning une clé de type INTEGER, car une date occupe un espace de stockage minime proche de celui de l'entier, voire moins.

Il faut ensuite remplir les tables avec les données de la table S_NEWS.TRANSPORS_NEWS.TSP.

La modification de la table S_NEWS.TRANSPORS_NEWS.TSP peut être ainsi effectuée :

1) Ajout de la colonne TRN_ID avec contrainte d'intégrité référentielle :

```
ALTER TABLE S_NEWS.TRANSPORS_NEWS.TSP
ADD COLUMN TRN_ID INTEGER NOT NULL DEFAULT 1
                  FOREIGN KEY REFERENCES S_NEWS.TOURNEE_TRN (TRN_ID)
```

Il convient de prévoir une valeur par défaut, arbitrairement choisie parmi les clés nouvellement insérées dans la table. En effet, l'obligation de valeur (NOT NULL) interdit que la colonne soit vide. Sans cette valeur par défaut arbitraire, un tel ordre de modification serait rejeté.

2) Il faut ensuite transférer les données en respectant la cohérence et l'intégrité de l'information. Un ordre SQL UPDATE (voir chapitre 6) permet d'assurer ce transfert :

```
UPDATE S_NEWS.TRANSPORS_NEWS.TSP
SET TRN_ID = (SELECT TRN_ID
              FROM   S_NEWS.TOURNEE_TRN TRN
              WHERE  TRN.TRN_TOURNEE = TSP_TOURNEE)
```

3) Suppression de la colonne TSP_TOURNEE devenue obsolète :

```
ALTER TABLE S_NEWS.TRANSPORS_NEWS.TSP
DROP COLUMN TSP_TOURNEE
```

4) Ajout de la colonne PLN_DATE avec contrainte d'intégrité référentielle :

```
ALTER TABLE S_NEWS.TRANSPORS_NEWS.TSP
ADD COLUMN PLN_DATE DATE NOT NULL DEFAULT CURRENS_NEWS.DATE
                    FOREIGN KEY REFERENCES S_NEWS.PLANNING_PLN (PLN_DATE)
```

5) Transfert des données de la colonne TSP_JOUR vers la colonne PLN_DATE à l'aide d'un ordre SQL UPDATE (voir chapitre 6) :

```
UPDATE S_NEWS.TRANSPORS_NEWS.TSP
SET PLN_DATE = TSP_JOUR
```

6) Suppression de la colonne TSP_JOUR devenue obsolète :

```
ALTER TABLE S_NEWS.TRANSPORS_NEWS.TSP
DROP COLUMN TSP_JOUR
```

EXERCICE 8 VALIDATION DES CONTRAINTES

Énoncé

Avec la table S_NEWS.FACTURE_FCT du schéma S_COM, définie comme suit :

```
CREATE TABLE S_COM.S_NEWS.FACTURE_FCT
(FCS_NEWS.NUMERO                INTEGER,
 FCS_NEWS.DATE_EMISSION         DATE    CONSTRAINT CURRENS_NEWS.DATE,
 FCS_NEWS.DATE_LIMITE_PAIEMENT  DATE    DEFAULT CURRENS_NEWS.DATE + 90 DAY);
```

Dans laquelle on insère les données suivantes à la date du 1er septembre 2008 (voir syntaxe de la commande INSERT au chapitre 6) :

```
INSERT INTO S_COM.S_NEWS.FACTURE_FCT
       VALUES (1, NULL, NULL);
INSERT INTO S_COM.S_NEWS.FACTURE_FCT
       VALUES (2, DEFAULT, DEFAULT);
INSERT INTO S_COM.S_NEWS.FACTURE_FCT
       VALUES (3, DEFAULT, NULL);
INSERT INTO S_COM.S_NEWS.FACTURE_FCT (FCS_NEWS.NUMERO)
       VALUES (4);
INSERT INTO S_COM.S_NEWS.FACTURE_FCT (FCS_NEWS.NUMERO, FCS_NEWS.DATE_EMISSION)
       VALUES (5, NULL);
INSERT INTO S_COM.S_NEWS.FACTURE_FCT (FCS_NEWS.NUMERO, FCS_NEWS.DATE_EMISSION)
       VALUES (6, DEFAULT);
INSERT          INTO          S_COM.S_NEWS.FACTURE_FCT          (FCS_NEWS.NUMERO,
FCS_NEWS.DATE_LIMITE_PAIEMENT)
       VALUES (7, NULL);
INSERT          INTO          S_COM.S_NEWS.FACTURE_FCT          (FCS_NEWS.NUMERO,
FCS_NEWS.DATE_LIMITE_PAIEMENT)
       VALUES (8, DEFAULT);
```

Indiquez le contenu de la table après insertion.

Solution

Les insertions suivantes seront rejetées :

```
FCS_NEWS.NUMERO  FCS_NEWS.DATE_EMISSION  FCS_NEWS.DATE_LIMITE_PAIEMENT
-----------  ------------------  ------------------------
1            NULL                NULL
2            2008-09-01          2008-11-30
3            2008-09-01          NULL
4            2008-09-01          2008-11-30
5            NULL                2008-11-30
6            2008-09-01          2008-11-30
7            2008-09-01          NULL
8            2008-09-01          2008-11-30
```

EXERCICE 9 VALIDATION DES CONTRAINTES

Énoncé

Soit la définition de table suivante :

```
CREATE TABLE S_NEWS.VOITURE_VTR
(VTR_ID              INTEGER   NOT NULL
                        PRIMARY KEY,
 VTR_IMMATRICUL      CHAR(10)  NOT NULL  UNIQUE
                        CHECK (VALUE = UPPER(VALUE)),
 VTR_CARBURANT       CHAR(2)   NOT NULL  DEFAULT 'ES'
                        CHECK (VALUE IN ('ES', 'GO', 'PL')),
 VTR_PUISSANCE_FISC  INTEGER   NOT NULL
                        CHECK (VALUE BETWEEN 1 AND 20),
 VTR_NB_PLACES       INTEGER   NOT NULL
                        CHECK (VALUE BETWEEN 1 AND 7),
 VTR_MODELE          VARCHAR(20)
                        CHECK (TRIM(BOTH, VALUE) <> ''),
 VTR_CONSTRUCTEUR    VARCHAR(16) NOT NULL
                        CHECK (TRIM(BOTH, VALUE) <> ''),
 VTR_NUMERO_SERIE    VARCHAR(25) NOT NULL
                        CHECK (TRIM(BOTH, VALUE) <> ''),
 CONSTRAINT          CK_IMMATRICULATION
                        CHECK (SUBSTRING(VTR_IMMATRICUL FROM 9 FOR 2)
                               BETWEEN '01' AND '95'
                            OR SUBSTRING(VTR_IMMATRICUL FROM 9 FOR 2)
                               IN ('2A', '2B')),
 CONSTRAINT          CK_PUISS_PLACE
                        CHECK (VTR_NB_PLACES - 1 < VTR_PUISSANCE_FISC),
 CONSTRAINT          UK_MDL_CTR_NSR UNIQUE
                        (VTR_MODELE, VTR_CONSTRUCTEUR, VTR_NUMERO_SERIE))
```

La fonction TRIM (BOTH... supprime les espaces en début et fin de chaîne. La fonction UPPER(...) met en majuscules une chaîne de caractères. La fonction SUBSTRING(... FROM... FOR...) extrait une chaîne de caractères le caractère de position "FROM" et pour un nombre de caractères défini par "FOR".

Parmi les lignes suivantes, lesquelles seront refusées et pourquoi ?

```
1)   VTR_ID             = 14
     VTR_IMMATRICUL     = '478 XDA 78'
     VTR_CARBURANT      = 'ES'
     VTR_PUISSANCE_FISC = 9
     VTR_NB_PLACES      = 5
     VTR_MODELE         = '305'
     VTR_CONSTRUCTEUR   = 'PEUGEOT'
     VTR_NUMERO_SERIE   = '00014578'

2)   VTR_ID             = 31
     VTR_IMMATRICUL     = '1447 MD 44'
     VTR_CARBURANT      = 'ES'
     VTR_NUMERO_SERIE   = 0001578
     VTR_PUISSANCE_FISC = '7'
     VTR_NB_PLACES      = 5
     VTR_MODELE         =
     VTR_CONSTRUCTEUR   = 'CITROËN'

 3)  VTR_ID             = 7
     VTR_NB_PLACES      = 4
     VTR_MODELE         = '204'
     VTR_CONSTRUCTEUR   = 'PEUGEOT'
     VTR_IMMATRICUL     = '5474 MRT 91'
     VTR_CARBURANT      = 'GO'
     VTR_PUISSANCE_FISC = 5
     VTR_NUMERO_SERIE   = '0001474578'
```

Énoncé *(suite)*

```
 4)  VTR_ID             = 11
     VTR_IMMATRICUL     = '1744 BC 76'
     VTR_CONSTRUCTEUR   =
     VTR_NUMERO_SERIE   = '00025687'
     VTR_MODELE         =
     VTR_CARBURANT      = 'GO'
     VTR_PUISSANCE_FISC = 7
     VTR_NB_PLACES      = 5

 5)  VTR_ID             = 15
     VTR_IMMATRICUL     = '4412 LR 75'
     VTR_CONSTRUCTEUR   = 'PEUGEOT'
     VTR_NB_PLACES      = 4
     VTR_MODELE         = '305'
     VTR_CARBURANT      = 'GO'
     VTR_PUISSANCE_FISC = 7
     VTR_NUMERO_SERIE   = '00014578'

 6)  VTR_ID             = 17
     VTR_CONSTRUCTEUR   = 'PEUGEOT'
     VTR_CARBURANT      =
     VTR_PUISSANCE_FISC = 7
     VTR_NB_PLACES      = 5
     VTR_IMMATRICUL     = '971 VTR 96'
     VTR_MODELE         = '306'
     VTR_NUMERO_SERIE   = '00017548'

 7)  VTR_ID             = 19
     VTR_PUISSANCE_FISC = 8
     VTR_NB_PLACES      = 5
     VTR_CONSTRUCTEUR   = 'MEGANE'
     VTR_IMMATRICUL     = '991 SDT 75'
     VTR_CARBURANT      = 'ES'
     VTR_MODELE         = 'RENAULT'
     VTR_NUMERO_SERIE   = '00014578'

 8)  VTR_ID             = 20
     VTR_CARBURANT      = 'ES
     VTR_PUISSANCE_FISC = 5
     VTR_IMMATRICUL     = '991 SDT 75'
     VTR_MODELE         = 'MEGANE'
     VTR_CONSTRUCTEUR   = 'RENAULT'
     VTR_NB_PLACES      = 4
     VTR_NUMERO_SERIE   = '00014578'

 9)  VTR_ID             = 14
     VTR_IMMATRICUL     = '4785 ZT 94'
     VTR_MODELE         =
     VTR_CONSTRUCTEUR   = 'RENAULT'
     VTR_CARBURANT      =
     VTR_PUISSANCE_FISC = 7
     VTR_NB_PLACES      = 5
     VTR_NUMERO_SERIE   = '0005784'

10)  VTR_ID             = 7
     VTR_IMMATRICUL     = '5474 MRT 91'
     VTR_PUISSANCE_FISC = 5
     VTR_NB_PLACES      = 4
     VTR_CARBURANT      = 'GPL'
     VTR_CONSTRUCTEUR   = 'CHRYSLER'
     VTR_MODELE         = 'PT CRUISER'
     VTR_NUMERO_SERIE   = '0000050214'
```

Solution

Les insertions suivantes seront rejetées :

4) La colonne 'VTR_CONSTRUCTEUR' est obligatoire du fait de la contrainte NOT NULL.

5) Il y a violation de la contrainte d'unicité UK_MDL_CTR_NSR : ce trio de valeurs existe déjà dans l'insertion réussie en 1).

6) Il y a violation de la contrainte de validation CK_IMMATRICULATION : les deux derniers caractères '96' n'entrent pas dans la liste des valeurs autorisées.

8) Il y a violation de la contrainte CK_PUISS_PLACE : le nombre de places doit être strictement inférieur à la puissance moins une unité.

9) Il y a violation de la clé primaire : la valeur 14 de la colonne VTR_ID a déjà été insérée en 1).

10) Il y a violation de la contrainte de validation sur la colonne VTR_CARBURANT : la valeur 'GPL', n'existe pas dans la liste des valeurs autorisées.

Par contre, l'insertion 2) doit réussir en dépit du fait que certaines expressions de valeurs ne sont pas du type attendu, car SQL effectue un transtypage implicite.

EXERCICE 10 VALIDATION DES CONTRAINTES

Énoncé

Soit les définitions de tables suivantes :

```
CREATE TABLE S_NEWS.PERSONNE_PRS
(PRS_ID         INTEGER NOT NULL PRIMARY KEY,
 PRS_NOM        CHAR(25) NOT NULL,
 PRS_PRENOM     VARCHAR(16))

CREATE TABLE S_NEWS.SERVICE_SVC
(SVC_ID         INTEGER NOT NULL PRIMARY KEY,
 SVC_LIBELLE    VARCHAR(16))

CREATE TABLE S_NEWS.EMPLOYEE_EMP
(PRS_ID         INTEGER NOT NULL PRIMARY KEY,
 PRS_SUPERIEUR  INTEGER FOREIGN KEY (PRS_ID)
                   REFERENCES S_NEWS.EMPLOYEE_EMP (PRS_ID),
 EMP_MATRICULE  CHAR(8) NOT NULL UNIQUE,
 SVC_ID         INTEGER NOT NULL FOREIGN KEY (SVC_ID)
                   REFERENCES S_NEWS.SERVICE_SVC (SVC_ID),
 CONSTRAINT     FK_PRS_EMP FOREIGN KEY (PRS_ID)
                   REFERENCES S_NEWS.PERSONNE_PRS (PRS_ID),
 CONSTRAINT     CK_PRS_EMP CHECK (PRS_ID <> PRS_SUPERIEUR))

CREATE TABLE S_NEWS.CLIENS_NEWS.CLI
(PRS_ID         INTEGER NOT NULL PRIMARY KEY,
 CLI_REMISE     FLOAT DEFAULT 0.0
                    CHECK (VALUE BETWEEN 0.0 AND 100.0),
 CONSTRAINT     FK_PRS_CLI FOREIGN KEY (PRS_ID)
                    REFERENCES S_NEWS.PERSONNE_PRS (PRS_ID))
```

Énoncé *(suite)*

et les données préalablement insérées :

```
S_NEWS.PERSONNE_PRS
PSR_ID    PRS_NOM     PRS_PRENOM
78        DUPONT      Jean
84        MARTIN      Pierre
11        LEBRUN      Paul
47        FAURE       Marc
21        LEVY        Adam

S_NEWS.SERVICE_SVC
SVC_ID    SVC_LIBELLE
7         COMMERCIAL
9         COMPTABILITE

S_NEWS.EMPLOYEE_EMP
PRS_ID    PRS_SUPERIEUR   EMP_MATRICULE   SVC_ID
11                        014578          7
78        21              074589          9
47        11              004587          7
21                        000112          9

S_NEWS.CLIENS_NEWS.CLI
PRS_ID     CLI_REMISE
84         7.5
21         30.0
```

Indiquez les lignes dont l'insertion ou la mise à jour seront refusées et dites pourquoi :

```
 1) S_NEWS.SERVICE_SVC :  SVC_ID        = 3
                          SVC_LIBELLE   = 'RESSOURCES HUMAINES'
 2) S_NEWS.SERVICE_SVC :  SVC_ID        = 5
                          SVC_LIBELLE   = 'PRODUCTION'
 3) S_NEWS.PERSONNE_PRS : PRS_ID        = 99
                          PRS_NOM       = 'DUBOIS'
                          PRS_PRENOM    = 'François'
 4) S_NEWS.PERSONNE_PRS : PRS_ID        = 32
                          PRS_NOM       = 'DURAND'
                          PRS_PRENOM    = 'Louis'
 5) S_NEWS.EMPLOYEE_EMP : EMP_MATRICULE = '004578'
                          PRS_ID        = 99
                          PRS_SUPERIEUR = 99
                          SVC_ID        = 5
 6) S_NEWS.EMPLOYEE_EMP : PRS_SUPERIEUR = 84
                          SVC_ID        = 5
                          EMP_MATRICULE = '004338'
                          PRS_ID        = 32
 7) S_NEWS.EMPLOYEE_EMP : PRS_ID        = 84
                          EMP_MATRICULE = '032578'
                          SVC_ID        = 6
                          PRS_SUPERIEUR = 47
 8) S_NEWS.CLIENS_NEWS.CLI   : CLI_ID        = 99
                          CLI_REMISE    = 10
 9) S_NEWS.CLIENS_NEWS.CLI   : CLI_ID        = 32
                          CLI_REMISE    = -10
10) S_NEWS.CLIENS_NEWS.CLI  :  CLI_ID        = 64
                          CLI_REMISE    = 0
```

Énoncé *(suite)*

```
11) S_NEWS.EMPLOYEE_EMP : PRS_ID        = 99
                          EMP_MATRICULE = '037721'
                          SVC_ID        = 5
                          PRS_SUPERIEUR = 84
12) S_NEWS.EMPLOYEE_EMP : PRS_ID        = 32
                          EMP_MATRICULE = '000787'
                          SVC_ID        = 3
                          PRS_SUPERIEUR =
```

Solution

Les lignes suivantes ne seront pas insérées :

1) La valeur de SVC_LIBELLE est trop longue.
5) Un employé ne peut pas être son propre supérieur (contrainte CHECK CK_PRS).
6) L'identifiant 84 relève d'une personne mais pas d'un salarié.
7) Le service d'identifiant 6 n'existe pas.
9) Une remise doit être comprise entre 0 et 100.
11) L'identifiant 84 relève d'une personne mais pas d'un salarié.

EXERCICE 11 MODIFICATION DE TABLE

Énoncé

Soit la définition de table suivante :

```
CREATE TABLE S_NEWS.FACTURE_FAC
(FAC_ID         INTEGER    NOT NULL PRIMARY KEY,
 FAC_DATE       DATE       NOT NULL DEFAULT CURRENS_NEWS.DATE,
 CLI_ID         INTEGER    NOT NULL,
 FAC_DATE_PMT   DATE,
 MDP_ID         INTEGER,
 CONSTRAINT     FK_FAC_CLI FOREIGN KEY (CLI_ID)
                   REFERENCES S_NEWS.CLIENS_NEWS.CLI (CLI_ID),
 CONSTRAINT     FK_FAC_MDP FOREIGN KEY (MDP_ID)
                   REFERENCES S_NEWS.MODE_PAIEMENS_NEWS.MDP (MDP_ID),
 CONSTRAINT     CK_DAS_NEWS.FAC CHECK (FAC_DATE_PMT >= FAC_DATE),
 CONSTRAINT     CK_PMS_NEWS.MPD CHECK
                  ((FAC_DATE_PMT IS NULL AND MDP_ID IS NULL)
                    OR NOT (FAC_DATE_PMT IS NULL OR MDP_ID IS NULL)))
```

Écrivez un script SQL capable de modifier le type des colonnes FAC_DATE_PMT et FAC_DTE en TIMESTAMP.

Solution

Voici l'ensemble des ordres SQL à passer pour réaliser ce changement de type :

```
-- ajout de colonnes temporaires
ALTER TABLE S_NEWS.FACTURE_FAC
   ADD COLUMN FAC_DATE_temp DATE
ALTER TABLE S_NEWS.FACTURE_FAC
   ADD COLUMN FAC_DATE_PMS_NEWS.temp DATE
-- bascule les données dans les colonnes temporaires
UPDATE S_NEWS.FACTURE_FAC SET FAC_DATE_temp = FAC_DATE
UPDATE S_NEWS.FACTURE_FAC SET FAC_DATE_PMS_NEWS.temp = FAC_DATE_PMT
```

```
-- supprime les contraintes liées aux colonnes visées
ALTER TABLE S_NEWS.FACTURE_FAC DROP CONSTRAINT CK_DAS_NEWS.FAC
ALTER TABLE S_NEWS.FACTURE_FAC DROP CONSTRAINT CK_PMS_NEWS.MPD
-- supprime la valeur par défaut de la colonne FAC_DATE
ALTER TABLE S_NEWS.FACTURE_FAC ALTER COLUMN FAC_DATE DROP DEFAULT
-- suppression des colonnes visées
ALTER TABLE S_NEWS.FACTURE_FAC DROP COLUMN FAC_DATE
ALTER TABLE S_NEWS.FACTURE_FAC DROP COLUMN FAC_DATE_PMT
-- ajout des colonnes avec le nouveau type
ALTER TABLE S_NEWS.FACTURE_FAC ADD COLUMN FAC_DATE TIMESTAMP
   NOT NULL DEFAULT CURRENT_TIMESTAMP
ALTER TABLE S_NEWS.FACTURE_FAC ADD COLUMN FAC_DATE_PMT TIMESTAMP
-- reprise des données
UPDATE S_NEWS.FACTURE_FAC SET FAC_DATE = FAC_DATE_temp
UPDATE S_NEWS.FACTURE_FAC SET FAC_DATE_PMT = FAC_DATE_PMS_NEWS.temp
-- ajout des contraintes précédemment supprimées
ALTER TABLE S_NEWS.FACTURE_FAC ADD CONSTRAINT CK_DAS_NEWS.FAC
            CHECK (FAC_DATE_PMT >= FAC_DATE)
ALTER TABLE S_NEWS.FACTURE_FAC ADD CONSTRAINT CK_PMS_NEWS.MPD CHECK
                  ((FAC_DATE_PMT IS NULL AND MDP_ID IS NULL)
                    OR NOT (FAC_DATE_PMT IS NULL OR MDP_ID IS NULL))
-- suppression des colonnes temporaires
ALTER TABLE S_NEWS.FACTURE_FAC DROP COLUMN FAC_DATE_temp
ALTER TABLE S_NEWS.FACTURE_FAC DROP COLUMN FAC_DATE_PMS_NEWS.temp
```

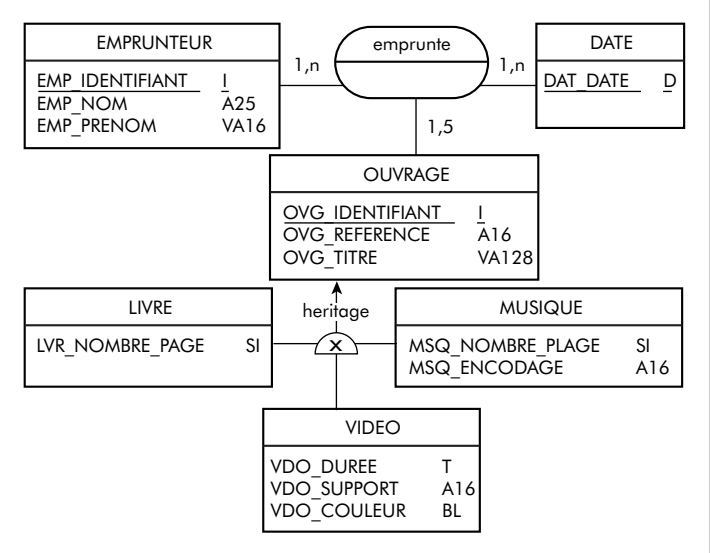
EXERCICE 12 CRÉATION D'UNE BASE DE DONNÉES

Énoncé

À partir des schémas conceptuels suivants (voir figures 3.3 et 3.4) d'une médiathèque, construisez les éléments de la base (table, vue, assertions, etc.) nécessaires à cette représentation.

Figure 3.3

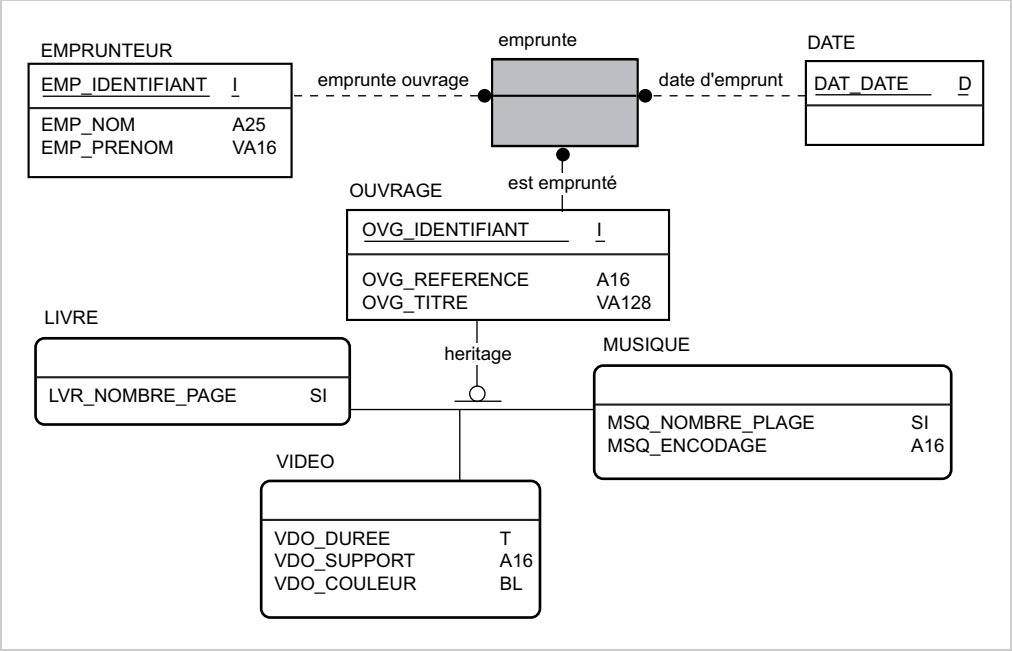
Modèle MERISE



Énoncé *(suite)*

Figure 3.1

Diagramme UML



Les règles de gestion spécifient qu'il n'est pas possible d'emprunter plus de cinq éléments à la fois, dont seulement deux supports musicaux ou vidéo.

Solution

Les tables sont les suivantes :

```
-- Table : S_NEWS.DATE_DAT
CREATE TABLE S_NEWS.DATE_DAT
(DAS_NEWS.DATE              DATE NOT NULL PRIMARY KEY)
-- Table : S_NEWS.EMPRUNTEUR_EMP
CREATE TABLE S_NEWS.EMPRUNTEUR_EMP
(EMP_IDENTIFIANT      INTEGER NOT NULL PRIMARY KEY,
 EMP_NOM              CHAR(25),
 EMP_PRENOM           VARCHAR(16))
-- Table : S_NEWS.OUVRAGE_OVG
CREATE TABLE S_NEWS.OUVRAGE_OVG
(OVG_IDENTIFIANT      INTEGER NOT NULL PRIMARY KEY,
 OVG_REFERENCE        CHAR(16) NOT NULL,
 OVG_TITRE            VARCHAR(128))
-- Table : S_NEWS.VIDEO_VDO
CREATE TABLE S_NEWS.VIDEO_VDO (
OVG_IDENTIFIANT       INTEGER NOT NULL PRIMARY KEY
   FOREIGN KEY REFERENCES S_NEWS.OUVRAGE_OVG (OVG_IDENTIFIANT),
VDO_DUREE             TIME,
VDO_SUPPORT           CHAR(16),
VDO_COULEUR           SMALLINT)
-- Table : S_NEWS.LIVRE_LVR
CREATE TABLE S_NEWS.LIVRE_LVR
(OVG_IDENTIFIANT      INTEGER NOT NULL PRIMARY KEY
```

```
    FOREIGN KEY REFERENCES S_NEWS.OUVRAGE_OVG (OVG_IDENTIFIANT),
 LVR_NOMBRE_PAGE      SMALLINT)
-- Table : S_NEWS.MUSIQUE_MSQ
CREATE TABLE S_NEWS.MUSIQUE_MSQ
(OVG_IDENTIFIANT      INTEGER NOT NULL PRIMARY KEY
    FOREIGN KEY REFERENCES S_NEWS.OUVRAGE_OVG (OVG_IDENTIFIANT),
 MSQ_NOMBRE_PLAGE     SMALLINT,
 MSQ_ENCODAGE         CHAR(16))
-- Table : TJ_EMPRUNTE_EPT
CREATE TABLE TJ_EMPRUNTE_EPT
(EMP_IDENTIFIANT      INTEGER NOT NULL
    FOREIGN KEY REFERENCES S_NEWS.EMPRUNTEUR_EMP (EMP_IDENTIFIANT),
 OVG_IDENTIFIANT      INTEGER NOT NULL
    FOREIGN KEY REFERENCES S_NEWS.OUVRAGE_OVG (OVG_IDENTIFIANT),
 DAS_NEWS.DATE             DATE NOT NULL
    FOREIGN KEY REFERENCES S_NEWS.DATE_DAT (DAS_NEWS.DATE),
 CONSTRAINT PK_EPT PRIMARY KEY
            (EMP_IDENTIFIANT, OVG_IDENTIFIANT, DAS_NEWS.DATE))
```

Les assertions sont les suivantes :

```
CREATE ASSERTION A_OVG_EXCLUSIF
CHECK NOT EXISTS (SELECT *
                  FROM   (SELECT OVG_IDENTIFIANT
                          FROM   S_NEWS.LIVRE_LVR
                          UNION  ALL
                          SELECT OVG_IDENTIFIANT
                          FROM   S_NEWS.MUSIQUE_MSQ
                          UNION  ALL
                          SELECT OVG_IDENTIFIANT
                          FROM   S_NEWS.VIDEO_VDO)
                  GROUP  BY OVG_IDENTIFIANT
                  HAVING COUNT(*) > 1 )
```

Cette assertion empêche, par exemple, qu'un livre ait le même identifiant qu'une vidéo (héritage exclusif).

```
CREATE ASSERTION A_EPS_NEWS.MAX5_OVG
CHECK NOT EXISTS (SELECT COUNT(*)
                  FROM   TJ_EMPRUNTE_EPT
                  GROUP  BY EMP_IDENTIFIANT, DAS_NEWS.DATE
                  HAVING COUNT(*) > 5)
```

Cette assertion limite les emprunts à cinq éléments.

```
CREATE ASSERTION A_EPS_NEWS.MAX2_MSQ_VDO
CHECK NOT EXISTS (SELECT COUNT(*)
                  FROM   TJ_EMPRUNTE_EPT
                  WHERE  OVG_IDENTIFIANT IN (SELECT OVG_IDENTIFIANT
                                             FROM   S_NEWS.MUSIQUE_MSQ
                                             UNION
                                             SELECT OVG_IDENTIFIANT
                                             FROM   S_NEWS.VIDEO_VDO)
                  GROUP  BY EMP_IDENTIFIANT, DAS_NEWS.DATE
                  HAVING COUNT(*) > 2)
```

Cette assertion limite à deux les emprunts de supports musicaux ou vidéo.

EXERCICE 13 CRÉATION D'UNE BASE DE DONNÉES

Énoncé

Créez les tables nécessaires à la commande de produits sur un site web. On y trouvera en particulier les éléments suivants : produits, clients, commande, paiement. Prévoyez des entiers pour toutes les clefs. Ajoutez les vues nécessaires à une consultation rapide des données, par exemple le total TTC et HT des lignes de commande et du total commande, le montant d'une remise éventuelle, etc.

Solution

Puisque le prix d'un même produit est susceptible d'évoluer dans le temps, deux méthodes de création des tables sont envisageables, suivant que l'on veut historiser les prix des produits ou bien que l'on désire geler les prix au moment de la commande.

1) avec les prix historisés :

```
-- la table des clients
CREATE TABLE S_NEWS.CLIENS_NEWS.CLI
(CLI_ID             INTEGER NOT NULL PRIMARY KEY,
 CLI_NOM            CHAR(25),
 CLI_MAIL           CHAR(128))
-- la table des produits
CREATE TABLE S_NEWS.PRODUIS_NEWS.PRD
(PRD_ID             INTEGER NOT NULL PRIMARY KEY,
 PRD_DESIGNATION    CHAR(32))
-- la table des taux de TVA
CREATE TABLE S_NEWS.TAUX_TVA_TVA
(TVA_ID             INTEGER NOT NULL PRIMARY KEY,
 TVA_LIBELLE        VARCHAR(25),
 TVA_TAUX           FLOAT NOT NULL, CHECK (VALUE > 0.0))
-- la table d'historisation des prix des produits
CREATE TABLE S_NEWS.PRIX_PRODUIS_NEWS.PPD
(PRD_ID             INTEGER NOT NULL
                       FOREIGN KEY
                       REFERENCES S_NEWS.PRODUIS_NEWS.PRD (PRD_ID),
 PPD_DATE           DATE NOT NULL DEFAULT CURRENS_NEWS.DATE,
 PPD_PRIX           DECIMAL(16,2),
 TVA_ID             INTEGER NOT NULL
                       FOREIGN KEY
                       REFERENCES S_NEWS.TAUX_TVA_TVA (TVA_ID),
 CONSTRAINT PK_PPD PRIMARY KEY (PRD_ID, PPD_DATE))
```

Bien que cette table soit parfaitement modélisée et respecte les formes normales, elle présente un inconvénient majeur : la difficulté de rechercher un prix à une date donnée. Il faut en effet parcourir dans l'ordre chronologique les lignes pour la référence du produit choisi, et s'arrêter sur la dernière ligne avant dépassement de la date à laquelle on commande le produit. Créer une vue simplifiant une telle recherche est donc tout indiqué. L'idée est alors de fournir un tarif par tranche de date, en assurant la continuité des dates entre le début et la fin de chaque tranche pour chaque tarif d'un même produit. Voici une telle vue :

```
-- la vue des prix des produits par intervalles de dates :
CREATE VIEW V_PRIX_PRODUIS_NEWS.PPR
AS
SELECT PRD.PRD_ID,
       PPD.PPD_DATE AS PPD_DATE_DEBUT,
       COALESCE(MIN(PPD2.PPD_DATE)
```

```
                - INTERVAL 1 DAY, CURREN_DATE) AS PPD_DATE_FIN,
       PRD.PRD_DESIGNATION,
       PPD.PPD_PRIX, PPD.TVA_ID, TVA.TVA_TAUX
FROM   S_NEWS.PRODUIS_NEWS.PRD PRD
       INNER JOIN S_NEWS.PRIX_PRODUIS_NEWS.PPD PPD
             ON PRD.PRD_ID = PPD.PRD_ID
       LEFT OUTER JOIN S_NEWS.PRIX_PRODUIS_NEWS.PPD PPD2
             ON PRD.PRD_ID = PPD2.PRD_ID
                AND PPD2.PPD_DATE > PPD.PPD_DATE
       INNER JOIN S_NEWS.TAUX_TVA_TVA TVA
             ON PPD.TVA_ID = TVA.TVA_ID
GROUP  BY PRD.PRD_ID, PRD.PRD_DESIGNATION,
          PPD.PPD_DATE, PPD.PPD_PRIX, PPD.TVA_ID,
          TVA.TVA_TAUX
-- la table des modes de paiement
CREATE TABLE S_NEWS.MODE_PAIEMENS_NEWS.MDP
(MDP_ID             INTEGER NOT NULL PRIMARY KEY,
 MDP_LIBELLE        VARCHAR(16))
-- la table des commandes
CREATE TABLE S_NEWS.COMMANDE_CMD
(CMD_ID             INTEGER NOT NULL PRIMARY KEY,
 CLI_ID             INTEGER NOT NULL
                       FOREIGN KEY
                       REFERENCES S_NEWS.CLIENS_NEWS.CLI (CLI_ID),
 CMD_DATE_COMMANDE  DATE NOT NULL DEFAULT CURRENS_NEWS.DATE,
 MDP_ID             INTEGER
                       FOREIGN KEY
                       REFERENCES S_NEWS.MODE_PAIEMENS_NEWS.MDP
                          (MDP_ID),
 CMD_DATE_PAIEMENT  DATE)
-- la table des lignes de commandes
CREATE TABLE S_NEWS.LIGNE_COMMANDE_LCD
(LCD_ID             INTEGER NOT NULL PRIMARY KEY,
 CMD_ID             INTEGER NOT NULL FOREIGN KEY
                       REFERENCES S_NEWS.COMMANDE_CMD (CMD_ID),
 PRD_ID             INTEGER NOT NULL FOREIGN KEY
                       REFERENCES S_NEWS.PRODUIS_NEWS.PRD (PRD_ID),
 CMD_QUANTITE       FLOAT NOT NULL DEFAULT 1.0,
 CMD_REMISE_POURCENT
                    FLOAT CHECK(VALUE BETWEEN 0.0 AND 100.0))
-- la vue des lignes de commande avec le calcul du prix payé HT et TTC,
-- le montant de la taxe et la remise :
CREATE VIEW V_LIGNE_COMMANDE_LCM
AS
SELECT LCD_ID, LCD.CMD_ID, LCD.PRD_ID, CMD_QUANTITE, CMD_REMISE_POURCENT,
       PPR.PPD_PRIX, PPR.TVA_TAUX,
       PPR.PPD_PRIX * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0) AS TOTAL_HT,
       PPR.PPD_PRIX * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0)
          * ( 1 + PPR.TVA_TAUX / 100.0 ) AS TOTAL_TTC,
       PPR.PPD_PRIX * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0)
          * PPR.TVA_TAUX / 100.0  AS TOTAL_TAXE,
       PPR.PPD_PRIX * CMD_QUANTITE
          * COALESCE(CMD_REMISE_POURCENT, 0) / 100.0
          * ( 1 + PPR.TVA_TAUX / 100.0 ) AS TOTAL_REMISE
FROM   S_NEWS.LIGNE_COMMANDE_LCD LCD
       INNER JOIN S_NEWS.COMMANDE_CMD CMD
             ON LCD.CMD_ID = CMD.CMD_ID
       INNER JOIN V_PRIX_PRODUIS_NEWS.PPR PPR
```

```
            ON LCD.PRD_ID = PPR.PRD_ID
               AND CMD.CMD_DATE_COMMANDE
                   BETWEEN PPD_DATE_DEBUT AND PPD_DATE_FIN
```

Notez que cette vue est basée sur la vue précédente et utilise le prédicat BETWEEN pour retrouver le bon prix de chaque produit à la date de la commande.

```
-- la vue de la commande avec le total HT et TTC, le montant global
-- de la remise et des taxes et le nombre de produits :
CREATE VIEW V_COMMANDE_CME
AS
SELECT CMD.CMD_ID, CLI_ID, CMD_DATE_COMMANDE, MDP_ID, CMD_DATE_PAIEMENT,
       SUM(CMD_QUANTITE) AS TOTAL_QUANTITE_COMMANDE,
       SUM(TOTAL_HT) AS TOTAL_HS_NEWS.COMMANDE,
       SUM(TOTAL_TTC) AS TOTAL_TTC_COMMANDE,
       SUM(TOTAL_TAXE) AS TOTAL_TAXE_COMMANDE,
       SUM(TOTAL_REMISE)  AS TOTAL_REMISE_COMMANDE
FROM   S_NEWS.COMMANDE_CMD CMD
       INNER JOIN V_LIGNE_COMMANDE_LCM LCM
             ON CMD.CMD_ID = LCM.CMD_ID
GROUP BY CMD.CMD_ID, CLI_ID, CMD_DATE_COMMANDE, MDP_ID, CMD_DATE_PAIEMENT
```

Là encore, la vue sur la commande est créée d'après la vue sur la ligne de commande, elle-même basée sur la vue des prix des produits.

2) avec les prix recopiés :

```
-- la table des clients
CREATE TABLE S_NEWS.CLIENS_NEWS.CLI
(CLI_ID             INTEGER NOT NULL PRIMARY KEY,
 CLI_NOM            CHAR(25),
 CLI_MAIL           CHAR(128))

-- la table des produits
CREATE TABLE S_NEWS.PRODUIS_NEWS.PRD
(PRD_ID             INTEGER NOT NULL PRIMARY KEY,
 PRD_DESIGNATION    CHAR(32),
 PRD_PRIX_UHT       DECIMAL (16,2) NOT NULL CHECK(VALUE > 0.0),
 PRD_TAUX_TVA       FLOAT NOT NULL DEFAULT 19.6)
-- la table des modes de paiement
CREATE TABLE S_NEWS.MODE_PAIEMENS_NEWS.MDP
(MDP_ID             INTEGER NOT NULL PRIMARY KEY,
 MDP_LIBELLE        VARCHAR(16))
-- la table des commandes
CREATE TABLE S_NEWS.COMMANDE_CMD
(CMD_ID             INTEGER NOT NULL PRIMARY KEY,
 CLI_ID             INTEGER NOT NULL
                       FOREIGN KEY
                       REFERENCES S_NEWS.CLIENS_NEWS.CLI (CLI_ID),
 CMD_DATE_COMMANDE  DATE NOT NULL DEFAULT CURRENS_NEWS.DATE,
 MDP_ID             INTEGER
                       FOREIGN KEY
                       REFERENCES S_NEWS.MODE_PAIEMENS_NEWS.MDP
                          (MDP_ID),
 CMD_DATE_PAIEMENT  DATE)
-- la table des lignes de commandes
CREATE TABLE S_NEWS.LIGNE_COMMANDE_LCD
(LCD_ID             INTEGER NOT NULL PRIMARY KEY,
 CMD_ID             INTEGER NOT NULL FOREIGN KEY
                       REFERENCES S_NEWS.COMMANDE_CMD (CMD_ID),
 PRD_ID             INTEGER NOT NULL FOREIGN KEY
                       REFERENCES S_NEWS.PRODUIS_NEWS.PRD (PRD_ID),
```

```
 CMD_QUANTITE       FLOAT NOT NULL DEFAULT 1.0,
 CMD_PRIX_UHT       FLOAT NOT NULL CHECK (VALUE > 0.0),
 CMD_TAUX_TVA       FLOAT NOT NULL DEFAULT 19.6 (CHECK VALUE > 0.0),
 CMD_REMISE_POURCENT
                    FLOAT CHECK(VALUE BETWEEN 0.0 AND 100.0))
-- la vue des lignes de commande avec le calcul du prix payé HT et TTC,
-- le montant de la taxe et la remise :
CREATE VIEW V_LIGNE_COMMANDE_LCM
AS
SELECT LCD_ID, CMD_ID, PRD_ID, CMD_QUANTITE,
       CMD_PRIX_UHT, CMD_TAUX_TVA, CMD_REMISE_POURCENT,
       CMD_PRIX_UHT * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0) AS TOTAL_HT,
       CMD_PRIX_UHT * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0)
          * ( 1 + CMD_TAUX_TVA / 100.0 ) AS TOTAL_TTC,
       CMD_PRIX_UHT * CMD_QUANTITE
          * (1 - COALESCE(CMD_REMISE_POURCENT, 0) / 100.0)
          * CMD_TAUX_TVA / 100.0  AS TOTAL_TAXE,
       CMD_PRIX_UHT * CMD_QUANTITE
          * COALESCE(CMD_REMISE_POURCENT, 0) / 100.0
          * ( 1 + CMD_TAUX_TVA / 100.0 ) AS TOTAL_REMISE
FROM   S_NEWS.LIGNE_COMMANDE_LCD
-- la vue de la commande avec le total HT et TTC, le montant global
-- de la remise et des taxes et le nombre de produits :
CREATE VIEW V_COMMANDE_CME
AS
SELECT CMD.CMD_ID, CLI_ID, CMD_DATE_COMMANDE, MDP_ID, CMD_DATE_PAIEMENT,
       SUM(CMD_QUANTITE) AS TOTAL_QUANTITE_COMMANDE,
       SUM(TOTAL_HT) AS TOTAL_HS_NEWS.COMMANDE,
       SUM(TOTAL_TTC) AS TOTAL_TTC_COMMANDE,
       SUM(TOTAL_TAXE) AS TOTAL_TAXE_COMMANDE,
       SUM(TOTAL_REMISE)  AS TOTAL_REMISE_COMMANDE
FROM   S_NEWS.COMMANDE_CMD CMD
       INNER JOIN V_LIGNE_COMMANDE_LCM LCM
             ON CMD.CMD_ID = LCM.CMD_ID
GROUP BY CMD.CMD_ID, CLI_ID, CMD_DATE_COMMANDE, MDP_ID, CMD_DATE_PAIEMENT
```

Notez dans les deux cas que cette dernière vue s'exprime de la même manière. Ceci prouve que les deux modèles sont interchangeables dans leurs finalités mais n'offrent pas les mêmes possibilités. Dans ce dernier modèle, nous ne pouvons que mesurer l'évolution des prix des marchandises achetées, tandis que le précédent montre aussi la courbe d'évolution des prix présentés.

Exercices

Sauf indication contraire, tous les exercices d'extraction de données portent sur la base exemple dont les schémas sont décrits au chapitre 1.
Les 16 premiers exercices traitent des fonctionnalités standard de l'instruction SELECT. Les suivants sont consacrés aux groupages multidimensionnels.

EXERCICE 1 FILTRAGE

Énoncé

Affichez les utilisateurs (numéro, nom, adresse e-mail) qui ne sont pas associés à une organisation.

Solution

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NULL OR TRIM(LEFT FROM USR_ORGANISATION) = ''
```

Outre l'absence de valeur caractérisée par le marqueur NULL, il ne faut pas oublier une chaîne de caractères vide.

Avec MS SQL Server, on écrira :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NULL OR LTRIM(USR_ORGANISATION) = ''
```

EXERCICE 2 FILTRAGE

Énoncé

Affichez les utilisatrices qui ne sont pas associées à une organisation.

Solution

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NULL
AND    USR_TITRE IN ('Mme', 'Mlle')
```

La contrainte CHECK sur la colonne USR_TITRE de la table T_UTILISATEUR_USR précise que les seuls titres autorisés sont : 'M.', 'Mme' et 'Mlle'.

EXERCICE 3 STATISTIQUE

Énoncé

Pour chaque identifiant de fournisseur d'accès Internet (FAI), donnez le nombre de serveurs attribués.

Solution

```
SELECT FAI_ID, COUNT(SRV_ID)
FROM   S_NEWS.T_SERVEUR_SRV
GROUP  BY FAI_ID
```

Le regroupement s'opère sur le numéro de fournisseur. Le comptage de serveur associé à chaque fournisseur se trouve dans la clause SELECT.

EXERCICE 4 STATISTIQUE

Énoncé

Donnez la taille en kilo-octets du plus gros fichier stocké dans la table S_NEWS.FICHIER_FIC.

Solution

```
SELECT  CAST(MAX(FIC_TAILLE_O) AS FLOAT) / 1024.0
FROM    S_NEWS.T_FICHIER_FIC
```

Les tailles de fichiers dans la table des news sont renseignées en octets. Pour calculer la taille en kilo-octets, il faut diviser cette information par 1 024 (kilobinaire). Cependant, FIC_TAILLE_O étant un entier, il convient de le transtyper en réel avant la division.

EXERCICE 5 STATISTIQUE ET FILTRAGE

Énoncé

Trouvez les identifiants des fournisseurs d'accès Internet (FAI) ayant un seul serveur.

Solution

```
SELECT FAI_ID
FROM   S_NEWS.T_SERVEUR_SRV
GROUP  BY FAI_ID
HAVING COUNT(SRV_ID)=1
```

Il faut tester le regroupement dans HAVING et non dans la clause WHERE (le prédicat porterait sur la table entière).

EXERCICE 6 CONSTRUCTION

Énoncé

Pour chaque utilisateur dont vous extrayez l'identifiant, le nom et le prénom, constituez un trigramme composé de la première lettre du prénom et des première et dernière lettres du nom, le tout en majuscules. À défaut de prénom, prenez les deux premières lettres du nom.

Solution

```
SELECT USR_ID,
       UPPER(CASE
             WHEN USR_PRENOM IS NULL
             THEN SUBSTRING(USR_NOM FROM 1 FOR 2)
               || SUBSTRING(USR_NOM FROM CHARACTER_LENGTH(USR_NOM) FOR 1)
             ELSE SUBSTRING(USR_PRENOM FROM 1 FOR 1)
               || SUBSTRING(USR_NOM FROM 1 FOR 1)
               || SUBSTRING(USR_NOM FROM CHARACTER_LENGTH(USR_NOM) FOR 1)
             END) AS TRIGRAMME,
       USR_NOM, USR_PRENOM
FROM   S_NEWS.T_UTILISATEUR_USR
```

EXERCICE 7 DÉCOMPOSITION

Énoncé

Pour chaque utilisateur dont vous extrayez l'identifiant et l'adresse e-mail, décomposez l'adresse e-mail en deux parties, avant et après l'arobase (@).

Solution

```
SELECT USR_ID, USR_MAIL,
       SUBSTRING(USR_MAIL FROM 1
                          FOR POSITION('@' IN USR_MAIL) -1)
          AS AVANS_NEWS.AD,
       SUBSTRING(USR_MAIL FROM POSITION('@' IN USR_MAIL) + 1
                          FOR CHARACTER_LENGTH(USR_MAIL)
                                 - POSITION('@' IN USR_MAIL))
          AS APRES_AD
FROM   S_NEWS.T_UTILISATEUR_USR
```

EXERCICE 8 RECHERCHE DE MOTIF

Énoncé

Quelles sont les news dont le GUID (identificateur global unique – colonne NEW_GLOBAL_ID) contient *au moins* deux fois le motif « 12 », et celles qui le contiennent *uniquement* deux fois ?

Solution

Contiennent au moins deux fois le motif « 12 » dans NEW_GLOBAL_ID :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_GLOBAL_ID LIKE '%12%12%'
```

Deux fois seulement :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_GLOBAL_ID LIKE '%12%12%'
  AND  NOT NEW_GLOBAL_ID LIKE '%12%12%12%'
```

ou :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_GLOBAL_ID LIKE '%12%12%'
  AND  NEW_GLOBAL_ID NOT LIKE '%12%12%12%'
```

Chapitre 4

EXERCICE 9 FILTRAGE

Énoncé

À partir de la base exemple, quelles sont les news qui n'ont pas obtenu de réponse ?

Solution

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_ID_PERE IS NULL
```

EXERCICE 10 FILTRAGE

Énoncé

Quelles sont les news postées au cours de la période allant du 31 janvier au 14 février 2005 (bornes incluses) ?

Solution

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT BETWEEN '2005-01-31'
                      AND '2005-02-14 23:59:59.999'
```

Il est important de noter que la borne haute de l'intervalle doit spécifier les heures, minutes, secondes et millièmes de secondes juste avant minuit, sinon aucune des news postées le 6 février 2005 ne sera récupérée. Voici une autre manière de procéder :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT >= '2005-01-31'
  AND  NEW_MOMENT <= '2005-02-15'
```

Ce qui suppose de connaître le jour suivant à moins qu'on ne le réalise de la sorte :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT >= '2005-01-31'
AND  NEW_MOMENT <  '2005-02-14' +
```

EXERCICE 11 FILTRAGE

Énoncé

Quelles sont les news postées au cours des dix-huit derniers mois ?

Solution

Une telle requête :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT >= CURRENT_TIMESTAMP - INTERVAL 18 MONTH
```

a toutes les chances d'échouer, bien que la plupart des SGBDR l'accepteraient. En effet, le nombre de jours de chaque mois est différent. SQL n'accepte de retirer ou d'ajouter des mois à une date que si les mois continus ont un même nombre de jours, ce qui n'arrive que dans deux cas : juillet-août et décembre-janvier. Pour répondre à une telle

demande, mieux vaut se fonder sur un nombre de jours « moyen », comme le nombre de jours comptable d'un mois (30) ou le nombre de jours moyen d'une période mensuelle (30,43685 jours) :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT >= CURRENT_TIMESTAMP - INTERVAL 6570 DAY
```

Ici, on a choisi le nombre de jours comptable (18 * 30 = 540). En cas d'utilisation d'un nombre moyen de jours, il faudrait faire la requête suivante :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_MOMENT >= CURRENT_TIMESTAMP
                     - INTERVAL 547 DAY 20 HOUR 43 MINUTE 9 SECOND
```

EXERCICE 12 TRI

Énoncé

Donnez la liste des utilisateurs triés par organisations et noms.

Solution

```
SELECT *
FROM   S_NEWS.T_UTILISATEUR_USR
ORDER  BY USR_ORGANISATION, USR_NOM
```

EXERCICE 13 TRI

Énoncé

Donnez la liste des news triées par dates.

Solution

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
ORDER  BY NEW_MOMENT DESC
```

La plupart du temps lorsqu'un tri est demandé sur une colonne de l'un des types DATE, TIME ou TIMESTAMP, on veut voir apparaître en premier les éléments les plus récents, d'où le tri descendant.

EXERCICE 14 TRI

Énoncé

Donnez la liste des utilisateurs triés par initiales.

Solution

```
SELECT SUBSTRING(USR_NOM, 1, 1),  SUBSTRING(USR_PRENOM, 1, 1), *
FROM   S_NEWS.T_UTILISATEUR_USR
ORDER  BY 1,2
```

Les initiales des utilisateurs n'existant pas dans les données, il faut faire en sorte de les « calculer ». Ici, nous avons choisi de créer ces initiales à l'aide de deux nouvelles colonnes

sans nom. Par conséquent, nous avons dû utiliser l'ordre positionnel des colonnes dans la clause SELECT. Une autre manière serait de concaténer les initiales. Mais, dans ce cas, il faut éviter d'avoir affaire à des valeurs absentes (NULL) :

```
SELECT SUBSTRING(USR_NOM, 1, 1) +
          COALESCE(SUBSTRING(USR_PRENOM, 1, 1), ''),
       *
FROM   S_NEWS.T_UTILISATEUR_USR
ORDER  BY 1,2
```

Certains SGBDR acceptent des tris externes, c'est-à-dire à base d'expressions qui ne figurent pas dans la clause SELECT :

```
SELECT *
FROM   S_NEWS.T_UTILISATEUR_USR
ORDER  BY SUBSTRING(USR_NOM, 1, 1) +
             COALESCE(SUBSTRING(USR_PRENOM, 1, 1), '')
```

Une autre manière serait de concaténer les initiales. Mais, dans ce cas, il faut éviter d'avoir affaire à des valeurs absentes (NULL) :

```
SELECT SUBSTRING(USR_NOM, 1, 1) +
          COALESCE(SUBSTRING(USR_PRENOM, 1, 1), ''),
       *
FROM   S_NEWS.T_UTILISATEUR_USR
ORDER  BY 1
```

EXERCICE 15 TRI

Énoncé

La table S_HOTEL.T_CHAMBRE_CHB regroupe les chambres d'un hôtel. Listez les chambres dans l'ordre physique des étages.

```
-- contenu de la table S_HOTEL.T_CHAMBRE_CHB
CHB_ID  CHB_NUMERO CHB_ETAGE CHB_COUCHAGE CHB_CONFORT
------- ---------- --------- ------------ ----------------------
1       01         RDC       2            Douche
2       02         RDC       3            Bain
3       03         RDC       4            Bain
4       10         1er       2            Douche
5       11         1er       3            Douche
6       12         1er       2            Bain
7       14         1er       3            Bain
8       20         2e        2            Douche
9       21         2e        3            Bain
10      22         2e        5            Bains
```

Solution

```
SELECT *, CASE CHB_ETAGE
             WHEN 'RDC' THEN 0
             WHEN '1er' THEN 1
             WHEN '2e ' THEN 2
          END AS ETAGE
FROM   S_HOTEL.T_CHAMBRE_CHB
ORDER  BY ETAGE
```

La colonne CHB_ETAGE ne permet en aucun cas d'effectuer un tri efficient, car c'est une colonne littérale qui exige un tri numérique. Il faut donc substituer à chaque étage une valeur numérique correspondant au tri souhaité. C'est la structure CASE qui réalise cette substitution à la volée.

À nouveau si votre SGBDR le supporte, vous pouvez opter pour un tri externe :

```
SELECT *
FROM   S_HOTEL.T_CHAMBRE_CHB
ORDER  BY CASE CHB_ETAGE
             WHEN 'RDC' THEN 0
             WHEN '1er' THEN 1
             WHEN '2e ' THEN 2
          END
```

EXERCICE 16 TRI

Énoncé

Donnez la liste des news par mois.

Solution

```
SELECT *, EXTRACT(YEAR  FROM NEW_MOMENT) AS Y,
          EXTRACT(MONTH FROM NEW_MOMENT) AS M
FROM   S_NEWS.T_NEWS_NEW
ORDER  BY Y DESC, M DESC
```

Attention de ne pas considérer le mois comme seul critère auquel cas les mois de janvier de toutes les années seraient confondus. Dans une telle demande, on sous-entend que le tri doit s'effectuer pour chaque mois de chaque année.

EXERCICE 17 STATISTIQUES AVANCÉES (OLAP)

Énoncé

Soit la table S_VENTE.T_CHAUSSURE_VCS qui fournit les quantités vendues par dates, modèles, tailles et couleurs :

```
VCS_DATE    VCS_MODELE    VCS_TAILLE  VCS_COULEUR VCS_QUANTITE
----------- ------------------------- ----------- ------------
2005-01-03  Nike Air 101  44          Blanc       2
2005-01-04  Adidas Star   45          Rouge       3
2005-01-04  Nike Air 101  43          Blanc       1
2005-01-04  Nike Air 101  44          Blanc       1
2005-01-03  Nike Air 101  43          Noir        1
2005-01-05  Nike Air 101  43          Rouge       1
2005-01-06  Adidas Star   47          Bleu        1
2005-01-05  Adidas Star   44          Blanc       3
2005-01-05  Adidas Star   45          Blanc       1
2005-01-06  Nike Air 101  43          Bleu        1
2005-01-07  Adidas Star   44          Noir        3
2005-01-08  Adidas Star   45          Bleu        3
2005-01-07  Nike Air 101  43          Bleu        1
2005-01-08  Adidas Star   44          Blanc       3
2005-01-08  Adidas Star   45          Bleu        3
2005-01-08  Nike Air 101  43          Bleu        2
```

Donnez les statistiques des ventes :

1)par modèles ;

2)par modèles et tailles ;

3)par modèles, tailles et couleurs.

Solution

1) Par modèles :

```
SELECT VCS_MODELE, SUM(VCS_QUANTITE ) AS NOMBRE
FROM   S.VENTE.T_CHAUSSURE_VCS
GROUP  BY ROLLUP(VCS_MODELE)


VCS_MODELE       NOMBRE
---------------- -----------
Adidas Star      20
Nike Air 101     10
NULL             30
```

Ici, un ROLLUP suffit, car nous n'avons qu'une seule dimension à explorer. Notez qu'une présentation astucieuse consisterait à remplacer le marqueur NULL par un littéral plus censé, à l'aide par exemple de la fonction GROUPING :

```
SELECT CASE
          WHEN GROUPING(VCS_MODELE) = 1 THEN 'TOUS'
          ELSE VCS_MODELE
       END AS MODELE,
       SUM(VCS_QUANTITE ) AS NOMBRE
FROM   S.VENTE.T_CHAUSSURE_VCS
GROUP  BY ROLLUP(VCS_MODELE)
ORDER  BY MODELE

MODELE           NOMBRE
---------------- -----------
Adidas Star      20
Nike Air 101     10
TOUS             30
```

2) Par modèles et tailles :

```
SELECT VCS_MODELE, VCS_TAILLE, SUM(VCS_QUANTITE ) AS NOMBRE
FROM   S.VENTE.T_CHAUSSURE_VCS
GROUP  BY CUBE(BY VCS_MODELE, VCS_TAILLE)
ORDER  BY VCS_MODELE, VCS_TAILLE

VCS_MODELE       VCS_TAILLE  NOMBRE
---------------- ----------- -----------
NULL             NULL        30
NULL             43          7
NULL             44          12
NULL             45          10
NULL             47          1
Adidas Star      NULL        20
Adidas Star      44          9
Adidas Star      45          10
Adidas Star      47          1
Nike Air 101     NULL        10
Nike Air 101     43          7
Nike Air 101     44          3
```

Il faut utiliser le CUBE, et non plus le ROLLUP, sous peine de voir disparaître des cumuls par tailles et/ou par modèles. Là encore, on pourrait substituer aux marqueurs NULL un littéral plus explicite, en faisant attention à la problématique de transtypage au regard du tri :

```
SELECT CASE
          WHEN VCS_MODELE IS NULL THEN 'TOUS'
          ELSE VCS_MODELE
```

```
       END AS MODELE,
       CASE
          WHEN VCS_TAILLE IS NULL THEN 'TOUTES'
          ELSE CAST(VCS_TAILLE AS VARCHAR(12))
       END AS TAILLE,
       SUM(VCS_QUANTITE ) AS NOMBRE
FROM  S.VENTE.T_CHAUSSURE_VCS
GROUP  BY CUBE(VCS_MODELE,  VCS_TAILLE)
ORDER  BY 1, 2

MODELE           TAILLE       NOMBRE
---------------- ------------ -----------
Adidas Star      44           9
Adidas Star      45           10
Adidas Star      47           1
Adidas Star      TOUTES       20
Nike Air 101     43           7
Nike Air 101     44           3
Nike Air 101     TOUTES       10
TOUS             43           7
TOUS             44           12
TOUS             45           10
TOUS             47           1
TOUS             TOUTES       30
```

3) Par modèles, tailles et couleurs :

```
SELECT CASE
          WHEN VCS_MODELE IS NULL THEN 'TOUS'
          ELSE VCS_MODELE
       END AS MODELE,
       CASE
          WHEN VCS_TAILLE IS NULL THEN 'TOUTES'
          ELSE CAST(VCS_TAILLE AS VARCHAR(12))
       END AS TAILLE,
       CASE
          WHEN VCS_COULEUR IS NULL THEN 'TOUTES'
          ELSE VCS_COULEUR
       END AS COULEUR,
       SUM(VCS_QUANTITE ) AS NOMBRE
FROM   S.VENTE.T_CHAUSSURE_VCS
GROUP  BY CUBE(VCS_MODELE, VCS_TAILLE, VCS_COULEUR)
ORDER  BY 1, 2
```

EXERCICE 18 STATISTIQUES AVANCÉES

Énoncé

Quels sont les six premiers utilisateurs dans l'ordre alphabétique des prénoms ?

Solution

Cette question n'est simple qu'en apparence, car elle pose le problème des valeurs absentes et redondantes. En l'occurrence, une première approche pour trouver les cinq premiers prénoms dans l'ordre alphabétique peut utiliser la requête suivante :

```
SELECT DISTINCT USR_NOM, USR_PRENOM,
        DENSE_RANK() OVER(ORDER BY USR_PRENOM) AS RANG
FROM    S_NEWS.T_UTILISATEUR_USR
```

```
WHERE   USR_PRENOM IS NOT NULL
ORDER   BY RANG

USR_NOM         USR_PRENOM       RANG
------------------------------- -------
VALLADON        Antoine          1
ENTE            Aurélia          2
MAILLANT        Catherine        3
DUPONT          Charles          4
CLERC           François         5
DUPONT          François         5
COWARD          John Leslie      6
MÜLLER          Marcel           7
LEROY           Olivier          8
LEROY           Émilie           9
DUVAL           Éric             10
PRUD'HOMME      Étienne          11
```

L'usage de la fonction DENSE_RANK est rendu nécessaire car certains prénoms sont présents plusieurs fois dans la table. Mais il n'est pas possible d'utiliser un filtre de rang, de telles fonctions s'appliquant après que les résultats de la requête ont été extraits. De fait, une telle syntaxe est interdite :

```
SELECT DISTINCT USR_NOM, USR_PRENOM,
       DENSE_RANK() OVER (ORDER BY USR_PRENOM) AS RANG
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  WHERE   USR_PRENOM IS NOT NULL
  AND  DENSE_RANK() OVER (ORDER BY USR_PRENOM) <= 5
ORDER  BY RANG
```

De la même manière, un filtre HAVING ne serait pas accepté, car les fonctions de fenêtrage n'opèrent en définitive qu'après que tous les résultats sont connus, y compris d'éventuels calculs d'agrégats.

Remarquez cependant l'ambiguïté levée par les utilisateurs DUPONT Charles qui sont au nombre de deux, et les François qui sont eux aussi plusieurs. De fait, s'arrêter au rang 6 dans ce cas permet de récupérer sept lignes !

Voici une autre façon de procéder qui se fonde sur la fonction d'énumération des lignes :

```
SELECT DISTINCT USR_NOM, USR_PRENOM,
       ROW_NUMBER() OVER(ORDER BY USR_PRENOM) AS RANG
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_PRENOM IS NOT NULL
ORDER  BY RANG

USR_NOM         USR_PRENOM      RANG
--------------- --------------------
VALLADON        Antoine         1
ENTE            Aurélia         2
MAILLANT        Catherine       3
DUPONT          Charles         4
DUPONT          Charles         5
CLERC           François        6
DUPONT          François        7
```

```
COWARD          John Leslie     8
MÜLLER          Marcel          9
LEROY           Olivier         10
LEROY           Émilie          11
DUVAL           Éric            12
PRUD'HOMME      Étienne         13
```

Mais le résultat est tout aussi ambigu, et fort difficile à exploiter dans sa logique sans admettre des *ex æquo* et prévoir l'apparition de lignes surnuméraires.

Nous verrons que les solutions à une telle demande passent par une requête imbriquée dans la clause FROM, que nous étudierons au chapitre 5 lorsque nous parlerons des sous-requêtes.

Certains SGBDR proposent des filtres portant sur le *nombre de lignes*. Voici un exemple pour MS SQL Server :

```
SELECT TOP 5 *
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_PRENOM IS NOT NULL
ORDER  BY USR_PRENOM
```

et pour MySQL :

```
SELECT *
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_PRENOM IS NOT NULL
ORDER  BY USR_PRENOM
LIMIT  5
```

Mais l'échantillon renvoyé est arbitraire ; il peut être différent si on exécute à nouveau la requête dans le temps (choix des doublons).

EXERCICE 19 FILTRE COMPLEXE

Énoncé

Donnez, dans l'ordre alphabétique, prénom+nom, la liste des utilisateurs qui viennent après Étienne PRUD'HOMME.

Solution

La solution passe par l'utilisation du constructeur de lignes valuées :

```
SELECT USR_PRENOM, USR_NOM
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  (USR_PRENOM, USR_NOM) > ('Étienne', 'PRUD''HOMME')
ORDER  BY USR_PRENOM, USR_NOM
```

À défaut, il faut utiliser son équivalence logique :

```
SELECT USR_PRENOM, USR_NOM
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_PRENOM > 'Étienne'
   OR  USR_PRENOM = 'Étienne' AND  USR_NOM > 'PRUD''HOMME'
ORDER  BY USR_PRENOM, USR_NOM
```

Chapitre 4

EXERCICE 20 MULTISET

Énoncé

La table suivante présente les « mains » des joueurs lors d'une partie de belote. La colonne BLT_MAIN est un MULTISET. Quel joueur possède la dame de pique ?

```
-- contenu de la table S_NEWS.BELOTE_BLT
BLT_JOUEUR BLT_MAIN
---------- ------------------------------------------------------------
Paul       MULTISET['Pique Roi',  'Coeur Valet',  'Trefle sept']
Jean       MULTISET['Carreau As', 'Carreau huit', 'Trefle dix']
Marc       MULTISET['Coeur Dame', 'Pique Dame',   'Trefle neuf']
Jack       MULTISET['Coeur Roi',  'Pique dix,     'Pique Valet']
```

Solution

Plusieurs solutions sont possibles. Soit on utilise le prédicat MEMBER OF, soit on construit un MULTISET avec la carte « Pique Dame » et on utilise le prédicat SUBMULTISET :

```
SELECT BLT_JOUEUR
FROM   S_NEWS.BELOTE_BLT
WHERE  'Pique Dame' MEMBER OF BLT_MAIN

SELECT BLT_JOUEUR
FROM   S_NEWS.BELOTE_BLT
WHERE  MULTISET ['Pique Dame'] SUBMULTISET BLT_MAIN
```

EXERCICE 21 MULTISET

Énoncé

La table suivante présente des mesures effectuées sur des machines. La colonne MSR_MESURE est un MULTISET. Extrayez la plus forte mesure pour la machine Rotor.

```
-- contenu de S_NEWS.MESURE_MSR
MSR_MACHINE MSR_MESURE
----------- --------------------------------------------------
Pompe       MULTISET[47.25, 54.13, 78.43, 11.26, 119.12])
Rotor       MULTISET[21.78, 45.96, 74.25, 23.21])
Moteur      MULTISET[35.58, 47.98, 36.74, 41.9, 9.98, 11.28])
```

Solution

La solution passe par la reconstruction du MULTISET propre au Rotor en tant que table (fonction UNNEST) d'où l'on extrait la valeur maximale :

```
SELECT MAX(MSS_MESURE)
```

```
FROM   UNNEST(SELECT MSR_MESURES
              FROM   S_NEWS.MESURE_MSR
              WHERE  MSR_MACHINE = 'Rotor')
         AS S_NEWS.MESURES_MSS (MSS_MESURE)
```

EXERCICE 22 FONCTION ANALYTIQUE

Énoncé

Avec la table S_NOTE.S_NEWS.EXAMEN_EXM, résumant les notes des élèves pour différentes matières :

```
EXM_NOM         EXM_MATIERE     EXM_NOTE
--------------- --------------- ---------------
DUMAS           MATH            16
DUMAS           ANGLAIS         18
DUMAS           PHYSIQUE        13
WEBER           MATH            8
WEBER           ANGLAIS         13
WEBER           PHYSIQUE        11
SMITH           MATH            11
SMITH           ANGLAIS         9
SMITH           PHYSIQUE        19
```

Présentez les données du relevé de note de la classe sous la forme suivante :

```
EXM_NOM  EXM_MATIERE EXM_NOTE MOYENNE_MATIERE MOYENNE_ELEVE RANG_MATIERE
-------- ----------- -------- --------------- ------------- ------------
```

La colonne MOYENNE_MATIERE calcule la moyenne des notes relative à la matière notée. La colonne MOYENNE_ELEVE calcule la moyenne des notes de chaque élève quelle que soit sa matière. La colonne RANG_MATIERE calcule le classement du premier au dernier dans chaque matière.

Solution

```
SELECT *,
       AVG(EXM_NOTE) OVER(PARTITION BY EXM_MATIERE) AS MOYENNE_MATIERE,
       AVG(EXM_NOTE) OVER(PARTITION BY EXM_NOM) AS MOYENNE_ELEVE,
       RANK() OVER(PARTITION BY EXM_MATIERE ORDER BY EXM_NOTE DESC) AS RANG_MATIERE
FROM   S_NEWS.EXAMEN_EXM
```

Pour réaliser une telle requête, il faut utiliser la fonction d'agrégation AVG pour calculer les moyennes en utilisant un groupage par partition successivement sur les matières puis les noms des élèves. Enfin, pour le classement, vous devez utiliser la fonction RANK en partitionnant par matière avec un ordonnancement sur la note de la plus forte à la plus faible (DESC).

Chapitre 4

EXERCICE 23 FONCTION ANALYTIQUE

Énoncé

Avec une table stockant les cours de Bourse des titres chaque jour et définie comme suit :

```
CREATE TABLE S_BOURSE.S_NEWS.COTATION_CTT
(CTS_NEWS.STE        VARCHAR(16),    -- titre côté en bourse
 CTS_NEWS.DATE       DATE,           -- date de cotation
 CTS_NEWS.AN         INT,            -- année boursière
 CTS_NEWS.MOIS       INT,            -- mois boursier (de 1 à 12)
 CTS_NEWS.SEMAINE    INT,            -- semaine boursière (de 1 à 53)
 CTS_NEWS.COURS_OUV  DECIMAL(16,2),  -- cours à l'ouverture du marché
 CTS_NEWS.COURS_MAX  DECIMAL(16,2),  -- cours le plus haut en journée
 CTS_NEWS.COURS_MIN  DECIMAL(16,2),  -- cours le plus bas en journée
 CTS_NEWS.COURS_CLO  DECIMAL(16,2),  -- cours à la clôture du marché
 CTS_NEWS.VOLUME     INT,            -- nombre de titres échangés
 CONSTRAINT     PK_CTT PRIMARY KEY (CTS_NEWS.STE, CTS_NEWS.DATE))
```

Calculez la moyenne glissante sur 5 jours des titres pour le mois de janvier 2008. On prendra pour référence la valeur du titre à la clôture.

Solution

```
SELECT   CTS_NEWS.STE, CTS_NEWS.DATE,
         AVG(CTS_NEWS.COURS_CLO)
            OVER(PARTITION BY CTS_NEWS.STE
                 ORDER BY CTS_NEWS.DATE ROWS 4 PRECEDING) AS MOYENNE5
FROM     S_BOURSE.S_NEWS.COTATION_CTT
```

La moyenne mobile à 5 jours est la moyenne des clôtures du jour en cours ainsi que des 4 jours précédents. On peut encore écrire cette requête avec une clause WINDOW :

```
SELECT   CTS_NEWS.STE, CTS_NEWS.DATE,
         AVG(CTS_NEWS.COURS_CLO) OVER(W_5SLIDE) AS MOYENNE5
FROM     S_BOURSE.S_NEWS.COTATION_CTT
WINDOW   W_5SLIDE AS (PARTITION BY CTS_NEWS.STE
                      ORDER BY CTS_NEWS.DATE ROWS 4 PRECEDING)
```

EXERCICE 24 FONCTION ANALYTIQUE

Énoncé

En utilisant la table de l'exercice précédent, calculez le cours le plus fort et le plus faible de la semaine et du mois en regard de chaque cotation journalière.

Solution

```
SELECT   CTS_NEWS.STE, CTS_NEWS.DATE,
         MAX(CTS_NEWS.COURS_MAX) OVER(W_SEMAINE) AS MAXSEMAINE,
         MIN(CTS_NEWS.COURS_MIN) OVER(W_SEMAINE) AS MINSEMAINE,
         MAX(CTS_NEWS.COURS_MAX) OVER(W_MOIS) AS MAXMOIS,
         MIN(CTS_NEWS.COURS_MIN) OVER(W_MOIS) AS MINMOIS
FROM     S_BOURSE.S_NEWS.COTATION_CTT
WINDOW   W_SEMAINE AS (PARTITION BY CTS_NEWS.AN, CTS_NEWS.SEMAINE),
         W_MOIS AS  (PARTITION BY CTS_NEWS.AN, CTS_NEWS.MOIS)
```

EXERCICE 25 FONCTION ANALYTIQUE

Énoncé

La table suivante (S_GEO.S_NEWS.CODE_POSTAL_CDP) contient les codes postaux de toutes les communes de France. Les développeurs d'un site Web vous demandent de réaliser une requête qui, après filtrage, permet de visualiser 25 lignes de cette table dans l'ordre du nom de ville, pour chaque département, ceci afin de présenter les données sur différentes pages Web (pagination par nombre de lignes fixes).

```
CDP_CODE   CDP_VILLE
---------- ----------------
83670      BARJOLS
31000      TOULOUSE
51300      VITRY LE FRANÇOIS
06000      NICE
...
```

Pour cela, aidez-vous d'une vue.

Solution

La requête :

```
SELECT *, SUBSTRING(CDP_CODE FROM 1 FOR 2) AS DPT,
       ROW_NUMBER() OVER(PARTITION BY SUBSTRING(CDP_CODE FROM 1 FOR 2)
                         ORDER BY CDP_VILLE, CDP_CODE) AS N
FROM   S_GEO.S_NEWS.CODE_POSTAL_CDP
```

Ne peut être filtrée directement, car les fonctions analytiques sont calculées après que le jeu de données eut été produit. C'est pourquoi il est nécessaire de passer par une vue (nous verrons aux chapitres suivants qu'il est possible d'utiliser une sous-requête dite table dérivée ou encore une expression de table dans ce cas de figure) :

```
CREATE VIEW S_GEO.S_NEWS.CODE_POSTAL_PAGINE_CPP
AS
   SELECT *, SUBSTRING(CDP_CODE FROM 1 FOR 2) AS DPT,
          ROW_NUMBER() OVER(PARTITION BY SUBSTRING(CDP_CODE FROM 1 FOR 2)
                            ORDER BY CDP_VILLE, CDP_CODE) AS N
   FROM   S_GEO.S_NEWS.CODE_POSTAL_CDP
```

Dès lors, on peut filtrer comme suit, par exemple pour avoir la quatrième page pour le département 24 :

```
SELECT CDP_CODE, CDP_VILLE
FROM   S_GEO.S_NEWS.CODE_POSTAL_PAGINE_CPP
WHERE  DPT = 24
  AND  N BETWEEN (4 - 1) * 25  + 1 AND 4 * 25
```

Cette requête peut être paramétrée comme suit (voir chapitre 7) :

```
SELECT CDP_CODE, CDP_VILLE
FROM   S_GEO.S_NEWS.CODE_POSTAL_PAGINE_CPP
WHERE  DPT = :d
  AND  N BETWEEN (:p - 1) * :n  + 1 AND :p * :n
```

Dans laquelle les paramètres d, p et n représentent respectivement le numéro de département (d), la page voulue (p) et le nombre de lignes exigé par page (n).

Cette requête serait plus exacte si le numéro de département était extrait comme suit :

```
CASE
   WHEN CDP_CODE <= 97000 THEN SUBSTRING(CDP_CODE FROM 1 FOR 2)
   ELSE SUBSTRING(CDP_CODE FROM 1 FOR 3)
END
```

En effet, les départements d'outre-mer sont référencés sur 3 positions. Dans ce cas, la définition de vue précédente devient :

```
CREATE VIEW S_GEO.S_NEWS.CODE_POSTAL_PAGINE_CPP
AS
   SELECT *,
          CASE
             WHEN CDP_CODE <= 97000 THEN SUBSTRING(CDP_CODE FROM 1 FOR 2)
             ELSE SUBSTRING(CDP_CODE FROM 1 FOR 3)
          END AS DPT,
          ROW_NUMBER()
             OVER(PARTITION BY
                     CASE
                        WHEN CDP_CODE <= 97000
                           THEN SUBSTRING(CDP_CODE FROM 1 FOR 2)
                        ELSE SUBSTRING(CDP_CODE FROM 1 FOR 3)
                     END
                  ORDER BY CDP_VILLE, CDP_CODE) AS N
   FROM   S_GEO.S_NEWS.CODE_POSTAL_CDP
```

EXERCICE 26 FONCTION ANALYTIQUE

Énoncé

Soit la table S_COM.S_NEWS.VENTE_VTE (VTE_DATE, VTE_PRODUIT, VTE_MONTANT). Calculez pour chaque produit et par années/mois le chiffre d'affaires de la période et le nombre de ventes, ainsi que leurs cumuls depuis l'origine.

Solution

```
SELECT VTE_PRODUIT,
       EXTRACT(YEAR FROM VTE_DATE)                  AS AN,
       EXTRACT(MONTH FROM VTE_DATE)                 AS MOIS,
       SUM(VTE_MONTANT) OVER(W_SUMCNT)              AS TOTAL_MONTANT,
       SUM(SUM(VTE_MONTANT)) OVER (W_CUMULS)        AS CUMUL_MONTANT,
       COUNT(ALL VTE_MONTANT) OVER(W_SUMCNT)        AS QUANTITE,
       COUNT(COUNT(ALL VTE_MONTANT)) OVER(W_CUMULS) AS CUMUL_QUANTITE
FROM   S_COM.S_NEWS.VENTE_VTE
WINDOW W_SUMCNT AS (PARTITION BY VTE_PRODUIT,
                                 EXTRACT(YEAR FROM VTE_DATE),
                                 EXTRACT(MONTH FROM VTE_DATE))
       W_CUMULS AS (ORDER BY VTE_PRODUIT UNBOUNDED PRECEDING)
```

Exercices

Sauf indication contraire, les exercices d'extraction de données provenant de plusieurs tables portent sur la base exemple décrite au chapitre 1.

EXERCICE 1 JOINTURE

Énoncé

Donnez les informations numéro, nom des utilisateurs et titre des news des utilisateurs qui ont déjà déposé une news.

Solution

```
SELECT USR.USR_ID, USR.USR_NOM, NEW.NEW_TITRE
FROM   S_NEWS.T_UTILISATEUR_USR USR
       JOIN S_NEWS.T_NEWS_NEW NEW
            ON USR.USR_ID = NEW.USR_ID
```

Une seule jointure interne est nécessaire entre les deux tables.

EXERCICE 2 JOINTURES

Énoncé

Donnez les informations nom d'utilisateur et sujet des forums visités par l'utilisateur François DUPONT.

Solution

```
SELECT USR.USR_NOM, FRM.FRM_SUJET
FROM   S_NEWS.T_FORUM_FRM FRM
       JOIN S_NEWS.T_NEWS_NEW NEW
            ON FRM.FRM_ID = NEW.FRM_ID
       JOIN S_NEWS.T_UTILISATEUR_USR USR
            ON NEW.USR_ID = USR.USR_ID
WHERE  USR_NOM = 'DUPONT'
  AND  USR_PRENOM = 'François' ;
```

Trois tables sont en jointure, il est donc nécessaire d'utiliser deux clauses JOIN.

EXERCICE 3 JOINTURES

Énoncé

Donnez les noms des fichiers qui décrivent les news qui ont reçu des réponses et qui ont été postées par l'utilisateur Éric DUVAL.

Solution

```
SELECT FIC.FIC_NOM, FIC.FIC_TAILLE_O
FROM   S_NEWS.T_FICHIER_FIC FIC
       JOIN S_NEWS.T_NEWS_NEW NEW
            ON FIC.NEW_ID = NEW.NEW_ID
            JOIN S_NEWS.T_NEWS_NEW NW2
                 ON NEW.NEW_ID_PERE = NW2.NEW_ID
       JOIN S_NEWS.T_UTILISATEUR_USR USR
            ON NEW.USR_ID = USR.USR_ID
WHERE  USR_NOM = 'DUVAL'
  AND  USR_PRENOM = 'Éric'
```

Notez l'indentation volontaire des différentes jointures. En fait, une requête est considérée comme logiquement construite lorsque la clause FROM et les différentes jointures qui la contiennent constituent une arborescence. Par facilité, un tel arbre peut être représenté par une indentation des niveaux positifs en relation avec leur niveau père.

EXERCICE 4 JOINTURE MULTICONDITIONNELLE

Énoncé

Recherchez dans les tables suivantes les couples de locataires habitant, dans le même immeuble, un appartement différent de même superficie.

```
S_IMO.T_APPARTEMENT_APT
APT_NUM     APT_IMEUBLE      APT_SURFACE   APT_ETAGE
----------- ---------------- ------------- -----------
1           Les Lilas        150.0         3
2           Les Lilas         50.0         4
3           Les Lilas        200.0         2
4           Les Lilas         50.0         5
5           Les Bleuets      250.0         1
6           Les Bleuets      250.0         2

S_IMO.T_LOCATAIRE_LCT
LCT_NUM     APT_NUM     LCT_NOM          LCT_PRENOM
----------- ----------- ---------------- ----------
101         2           MARTIN           Michel
102         3           MAUREL           Didier
103         5           MEUNIER          Marie
104         6           MONOD            Marcel
105         6           MONOD            Jeannine
106         1           MOULIN           Éric
107         1           MOULIN           Martine
108         4           MEYER            Paul
```

Solution

```
SELECT DISTINCT LC1.LCT_PRENOM, LC1.LCT_NOM, LC2.LCT_PRENOM, LC2.LCT_NOM,
       AP1.APT_NUM, AP2.APT_NUM
FROM   S_IMO.T_LOCATAIRE_LCT LC1
       INNER JOIN S_IMO.T_APPARTEMENT_APT AP1
             ON LC1.APT_NUM = AP1.APT_NUM
       CROSS JOIN
       S_IMO.T_LOCATAIRE_LCT LC2
       INNER JOIN S_IMO.T_APPARTEMENT_APT AP2
             ON LC2.APT_NUM = AP2.APT_NUM
WHERE  AP1.APT_NUM <> AP2.APT_NUM
  AND  LC1.LCT_NUM <> LC2.LCT_NUM
```

```
AND  AP1.APT_SURFACE = AP2.APT_SURFACE
AND  AP1.APT_IMEUBLE = AP2.APT_IMEUBLE
AND  LC1.LCT_NUM > LC2.LCT_NUM
```

Cette solution consiste à lier les couples des tables T_APPARTEMENT_APT et T_LOCATAIRE_LCT deux fois et à opérer un produit cartésien sur les jointures ainsi réalisées. Il faut ensuite filtrer dans la clause WHERE les conditions de la requête :

1) Les appartements et les locataires ne doivent pas être les mêmes :

```
AP1.APT_NUM <> AP2.APT_NUM
LC1.LCT_NUM <> LC2.LCT_NUM
```

2) La surface et l'immeuble doivent être les mêmes :

```
AP1.APT_SURFACE = AP2.APT_SURFACE
AP1.APT_IMEUBLE = AP2.APT_IMEUBLE
```

Mais ces deux conditions ne suffisent pas. En effet, on se retrouve avec des « pseudo doublons » du fait que les tables sont présentes deux fois dans la jointure :

```
LCT_PRENOM LCT_NOM    APT_NUM     LCT_PRENOM LCT_NOM    APT_NUM
---------- ---------- ----------- ---------- ---------- -----------
Jeannine   MONOD      6           Marie      MEUNIER    5
Marcel     MONOD      6           Marie      MEUNIER    5
Marie      MEUNIER    5           Jeannine   MONOD      6
Marie      MEUNIER    5           Marcel     MONOD      6
Michel     MARTIN     2           Paul       MEYER      4
Paul       MEYER      4           Michel     MARTIN     2
```

Ici, les lignes 1 et 3 donnent la même information ! Il faut donc éliminer la moitié des lignes, c'est le sens de la dernière clause de filtrage :

```
LC1.LCT_NUM > LC2.LCT_NUM
```

Une autre façon de faire consiste à placer toutes les conditions de filtrage dans les clauses de jointure :

```
SELECT DISTINCT LC1.LCT_PRENOM, LC1.LCT_NOM, AP1.APT_NUM, LC2.LCT_PRENOM,
       LC2.LCT_NOM, AP2.APT_NUM
FROM   S_IMO.T_LOCATAIRE_LCT LC1
       INNER JOIN S_IMO.T_APPARTEMENT_APT AP1
             ON LC1.APT_NUM = AP1.APT_NUM
       INNER JOIN S_IMO.T_LOCATAIRE_LCT LC2
             ON LC1.LCT_NUM <> LC2.LCT_NUM
                AND LC1.LCT_NUM > LC2.LCT_NUM
       INNER JOIN S_IMO.T_APPARTEMENT_APT AP2
             ON LC2.APT_NUM = AP2.APT_NUM
                AND  AP1.APT_NUM <> AP2.APT_NUM
                AND  AP1.APT_SURFACE = AP2.APT_SURFACE
                AND  AP1.APT_IMEUBLE = AP2.APT_IMEUBLE
```

Mais cette requête est moins lisible.

EXERCICE 5 SOUS-REQUÊTE OU JOINTURE

Énoncé

Donnez le numéro et le nom du fichier le plus volumineux de la table S_NEWS.FICHIER_FIC.

Solution

```
SELECT  FIC_ID, FIC_NOM
FROM    S_NEWS.T_FICHIER_FIC
WHERE   FIC_TAILLE_O = (SELECT MAX(FIC_TAILLE_O)
                        FROM   S_NEWS.FICHIER_FIC)
```

Vous ne pouvez pas en un seul appel de table réaliser cette extraction. Elle nécessite de grouper des données (afin de calculer l'agrégat maximal sur la taille de fichier) et d'obtenir des informations sur le fichier en question.

Une autre solution consiste à utiliser une jointure croisée :

```
SELECT  FIC.FIC_ID, FIC.FIC_NOM
FROM    S_NEWS.T_FICHIER_FIC FIC
        CROSS JOIN S_NEWS.T_FICHIER_FIC FC2
GROUP   BY FIC.FIC_ID, FIC.FIC_NOM, FIC.FIC_TAILLE_O
HAVING  FIC.FIC_TAILLE_O = MAX(FC2.FIC_TAILLE_O)
```

Dans ce cas, on réalise le produit cartésien de la table sur elle-même, puis on filtre l'une des tables en considérant que la taille du fichier doit être la plus grande.

EXERCICE 6 MULTITABLE

Énoncé

Donnez le nom des utilisateurs (numéro, nom et adresse e-mail) dont l'organisation est aussi fournisseur de services Internet.

Solution

Aucune de ces requêtes ne traite totalement le problème :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION
       IN (SELECT FAI_NOM
           FROM S_NEWS.T_FOURNINES_NEWS.FAI)
```

Cette première formulation néglige les organisations pour lesquelles le nom de fournisseur d'accès n'est pas écrit dans la même casse (majuscules et minuscules). Pour éliminer cet effet de bord, nous pouvons écrire :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  UPPER(USR_ORGANISATION)
       IN (SELECT UPPER(FAI_NOM)
           FROM S_NEWS.T_FOURNINES_NEWS.FAI)
```

Cette nouvelle formulation ne donne pas non plus le résultat escompté. En effet, le nom du fournisseur d'accès peut se trouver n'importe où à l'intérieur de la colonne USR_ORGANISATION. Pour être plus précis, il faudrait élaborer la requête de la sorte :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
             ON UPPER(USR.USR_ORGANISATION)
                   LIKE '%' || UPPER(TRIM(BOTH FROM FAI.FAI_NOM)) || '%'
```

Notez que nous avons changé le IN en jointure ; cela revient au même, mais c'est plus propre. Cette formulation reste incomplète. En effet, le nom du fournisseur d'accès pourrait être une partie du nom de l'organisation.

Pour faire le rapprochement uniquement sur le mot constituant le nom du FAI, il faut entourer le nom de caractères blancs :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
             ON USR.USR_ORGANISATION LIKE
                   '% ' || UPPER(FAI.FAI_NOM) || ' %'
```

Cependant, le type de données du nom du FAI étant CHAR, des blancs de complément sont présents dans la concaténation avec le caractère '%'. Il faut éliminer ces blancs à l'aide de la fonction TRIM :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
             ON USR.USR_ORGANISATION LIKE
                   '% ' || UPPER(TRIM(RIGHT FROM FAI.FAI_NOM)) || ' %'
```

Mais cela n'est toujours pas suffisant pour traiter tous les cas de figure. En effet, le mot correspondant au nom du FAI peut se trouver au début ou à la fin de la chaîne USR_ORGANISATION (effets de bord). Dans ce cas, la présence de blancs dans le motif recherché empêchera de retrouver ces occurrences. Il faut donc ajouter la recherche de ces deux motifs, ce qui donne la solution complète que voici :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
             ON USR.USR_ORGANISATION LIKE
                   '% ' || UPPER(TRIM(RIGHT FROM FAI.FAI_NOM)) || ' %'
                OR USR.USR_ORGANISATION LIKE
                   '% ' || UPPER(TRIM(RIGHT FROM FAI.FAI_NOM))
                OR USR.USR_ORGANISATION LIKE
                   UPPER(TRIM(RIGHT FROM FAI.FAI_NOM)) || ' %'
```

EXERCICE 7 REQUÊTE ENSEMBLISTE

Énoncé

Trouvez les organisations qui sont aussi fournisseurs de services Internet.

Solution

```
SELECT USR_ORGANISATION
FROM   S_NEWS.T_UTILISATEUR_USR
INTERSECT
SELECT FAI_NOM
FROM   S_NEWS.T_FOURNINES_NEWS.FAI
```

Si votre SGBDR n'implémente pas l'intersection SQL, vous pouvez utiliser une jointure :

```
SELECT USR_ORGANISATION
FROM   S_NEWS.T_UTILISATEUR_USR
       INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI
             ON USR_ORGANISATION = FAI_NOM
```

ou une sous-requête :

```
SELECT USR_ORGANISATION
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IN ( SELECT FAI_NOM
                             FROM   S_NEWS.T_FOURNINES_NEWS.FAI )
```

```
SELECT USR_ORGANISATION
FROM   S_NEWS.T_UTILISATEUR_USR USR
WHERE  EXISTS (SELECT *
               FROM   S_NEWS.T_FOURNINES_NEWS.FAI FAI
               WHERE  USR.USR_ORGANISATION = FAI.FAI_NOM)
```

Si l'intersection porte sur plusieurs colonnes, il convient d'utiliser le ROW VALUE CONSTRUCTOR (constructeur de lignes valuées) avec le prédicat IN, ou opérer à l'aide d'une jointure, ou encore utiliser le prédicat EXISTS.

EXERCICE 8 REQUÊTE ENSEMBLISTE

Énoncé

Donnez la liste des organisations, des fournisseurs d'accès à Internet et des forums, en les classant par ordre alphabétique.

Solution

```
SELECT DISTINCT 'Organisation', USR_ORGANISATION, 1
FROM  S_NEWS.T_UTILISATEUR_USR
UNION
SELECT 'Fournisseur d''accès', FAI_NOM, 2
FROM S_NEWS.T_FOURNINES_NEWS.FAI
UNION
SELECT 'Forum', FRM_NOM, 3
FROM   S_NEWS.T_FORUM_FRM
ORDER BY 3, 2
```

Il est nécessaire d'ajouter une colonne dont la valeur servira uniquement au tri imposé, car le classement organisations, puis fournisseurs d'accès, puis forums ne correspond ni à l'ordre alphabétique ascendant, ni à celui descendant.

Notons qu'il est plus conforme d'identifier les colonnes par un nom :

```
SELECT DISTINCT 'Organisation' AS NATURE,
       USR_ORGANISATION AS NOM, 1 AS ORDRE
FROM   S_NEWS.T_UTILISATEUR_USR
UNION
SELECT 'Fournisseur d''accès', FAI_NOM, 2
FROM   S_NEWS.T_FOURNINES_NEWS.FAI
UNION
SELECT 'Forum', FRM_NOM, 3
FROM   S_NEWS.T_FORUM_FRM
ORDER  BY ORDRE, NOM
```

Si votre SGBDR n'implémente pas l'opérateur d'union SQL, vous pouvez utiliser un FULL OUTER JOIN entre les tables, combiné à la fonction COALESCE ou à la structure CASE :

```
SELECT DISTINCT
       CASE
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN 'Forum'
          ELSE 'Fournisseur d''accès'
          END
       ELSE 'Organisation'
       END,
```

```
       CASE
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN FRM_NOM
          ELSE FAI_NOM
          END
       ELSE USR_ORGANISATION
       END,
       CASE
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN 3
          ELSE 2
          END
       ELSE 1
       END
FROM  S_NEWS.T_UTILISATEUR_USR
      FULL OUTER JOIN S_NEWS.T_FOURNINES_NEWS.FAI
           ON USR_ORGANISATION = FAI_NOM
      FULL OUTER JOIN S_NEWS.T_FORUM_FRM
           ON USR_ORGANISATION = FRM_NOM
ORDER BY 3, 2
```

Il faut cependant avoir l'assurance :

•qu'aucun nom d'organisation n'est un nom de FAI ou de forum ;

•qu'aucun nom n'est NULL.

Dans le cas contraire, on peut écrire la jointure comme ceci :

```
FULL OUTER JOIN S_NEWS.T_FOURNINES_NEWS.FAI
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FAI_NOM, '') + '2'
      FULL OUTER JOIN S_NEWS.T_FORUM_FRM
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FRM_NOM, '') + '3'
```

En fait, chacune de ces requêtes pose un léger problème. La première fait apparaître un NULL au niveau des organisations (ce qui est logique car certains utilisateurs n'ont pas d'organisation renseignée) et la seconde, un NULL dans le nom des forums, ce qui peut paraître étrange mais est dû au NULL dans organisation, qui, du fait de la jointure externe bilatérale, est repoussé dans la dernière table jointe.

Voici ces requêtes rectifiées pour que tous les noms soient valués :

```
SELECT DISTINCT 'Organisation' AS NATURE,
       USR_ORGANISATION AS NOM, 1 AS ORDRE
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NOT NULL
UNION
SELECT 'Fournisseur d''accès', FAI_NOM, 2
FROM   S_NEWS.T_FOURNINES_NEWS.FAI
WHERE  FAI_NOM IS NOT NULL
UNION
SELECT 'Forum', FRM_NOM, 3
FROM   S_NEWS.T_FORUM_FRM
WHERE  FRM_NOM IS NOT NULL
ORDER  BY ORDRE, NOM
```

ou

```
SELECT DISTINCT
       CASE
```

```
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN 'Forum'
          ELSE 'Fournisseur d''accès'
          END
       ELSE 'Organisation'
       END AS NATURE,
       CASE
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN FRM_NOM
          ELSE FAI_NOM
          END
       ELSE USR_ORGANISATION
       END AS NOM,
       CASE
       WHEN USR_ORGANISATION IS NULL THEN
          CASE
          WHEN FAI_NOM IS NULL THEN 3
          ELSE 2
          END
       ELSE 1
       END AS ORDRE

FROM  S_NEWS.T_UTILISATEUR_USR
      FULL OUTER JOIN S_NEWS.T_FOURNINES_NEWS.FAI
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FAI_NOM, '') + '2'
      FULL OUTER JOIN S_NEWS.T_FORUM_FRM
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FRM_NOM, '') + '3'
WHERE USR_ORGANISATION IS NOT NULL
      OR FAI_NOM IS NOT NULL
      OR FRM_NOM IS NOT NULL
ORDER BY ORDRE, NOM
```

Autre écriture plus condensée à l'aide de la fonction COALESCE :

```
SELECT DISTINCT
       CASE
          WHEN USR_ORGANISATION IS NOT NULL THEN 'Organisation'
          WHEN FAI_NOM IS NOT NULL THEN 'Fournisseur d''accès'
          ELSE 'Forum'
       END AS NATURE,
       COALESCE(USR_ORGANISATION, FRM_NOM, FAI_NOM) AS NOM,
       CASE
          WHEN USR_ORGANISATION IS NOT NULL THEN 1
          WHEN FAI_NOM IS NOT NULL THEN 2
          ELSE 3
       END AS ORDRE
FROM  S_NEWS.T_UTILISATEUR_USR
      FULL OUTER JOIN S_NEWS.T_FOURNINES_NEWS.FAI
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FAI_NOM, '') + '2'
      FULL OUTER JOIN S_NEWS.T_FORUM_FRM
           ON COALESCE(USR_ORGANISATION, '') + '1'
              = COALESCE(FRM_NOM, '') + '3'
WHERE USR_ORGANISATION IS NOT NULL
      OR FAI_NOM IS NOT NULL
      OR FRM_NOM IS NOT NULL
ORDER BY ORDRE, NOM
```

EXERCICE 9 REQUÊTE ENSEMBLISTE

Énoncé

Utilisez l'opérateur EXCEPT de différence ensembliste, afin d'afficher les utilisateurs masculins (numéro, nom, adresse e-mail) qui n'appartiennent pas à une organisation.

Solution

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NULL
EXCEPT
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_TITRE IN ('Mlle.', 'Mme.')
```

Autres possibilités :

```
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
EXCEPT
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_TITRE IN ('Mlle.', 'Mme.')
   OR  USR_ORGANISATION IS NOT NULL

SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_TITRE = 'M.'
EXCEPT
SELECT USR_ID, USR_NOM, USR_MAIL
FROM   S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ORGANISATION IS NOT NULL
```

EXERCICE 10 FILTRAGE

Énoncé

Quelles sont les news restées sans réponse ou sans suite ?

Solution

```
SELECT NEW_ID, NEW_TITRE, NEW_TEXTE
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_ID NOT IN (SELECT NEW_ID_PERE
                      FROM   S_NEWS.T_NEWS_NEW
                      WHERE  NEW_ID_PERE IS NOT NULL)
```

Notez la présence du filtre WHERE dans la sous-requête. Sans cela, des marqueurs NULL apparaîtraient dans la liste des résultats, ce qui entacherait le prédicat IN de la valeur UNKNOWN.

Autre solution possible, avec une différence :

```
SELECT NEW_ID, NEW_TITRE, NEW_TEXTE
FROM   S_NEWS.T_NEWS_NEW
EXCEPT
SELECT NEW_ID_PERE, NEW_TITRE, NEW_TEXTE
FROM   S_NEWS.T_NEWS_NEW
```

Chapitre 5

EXERCICE 11 AGRÉGATS IMBRIQUÉS

Énoncé

Calculez le nombre de fournisseurs d'accès Internet ayant un seul serveur attribué.

Solution

```
SELECT COUNT(FAI_ID)
FROM   (SELECT FAI_ID
        FROM   S_NEWS.T_SERVEUR_SRV
        GROUP  BY FAI_ID
        HAVING COUNT(SRV_ID) = 1)
```

Pour réaliser ce comptage, vous devez dans un premier temps trouver les FAI n'ayant qu'un seul serveur. C'est le sens de cette requête :

```
SELECT FAI_ID
FROM   S_NEWS.T_SERVEUR_SRV
GROUP  BY FAI_ID
HAVING COUNT(SRV_ID) = 1
```

Ensuite, en réintégrant cette requête dans la clause FROM d'une requête externe (à la manière d'une table), vous pouvez appliquer un nouvel agrégat pour compter les occurrences.

EXERCICE 12 AGRÉGATS IMBRIQUÉS

Énoncé

Donnez le nom du forum ayant reçu le plus grand nombre de news.

Solution

```
SELECT FRM_NOM
FROM   S_NEWS.T_FORUM_FRM FRM
       INNER JOIN S_NEWS.T_NEWS_NEW NEW
             ON FRM.FRM_ID = NEW.FRM_ID
GROUP BY FRM_NOM, FRM.FRM_ID
HAVING COUNT(*)  = ( SELECT MAX(NOMBRE)
                     FROM   ( SELECT COUNT(*) AS NOMBRE
                              FROM S_NEWS.T_NEWS_NEW
                              GROUP BY FRM_ID) )
```

Pour réaliser cette extraction, vous devez commencer par calculer le nombre de news par forums, à l'aide de cette requête :

```
SELECT COUNT(*) AS NOMBRE
FROM S_NEWS.T_NEWS_NEW
GROUP BY FRM_ID
```

Ensuite, encapsulez cette requête dans une requête recherchant le « maximum » sur l'ensemble des résultats :

```
SELECT MAX(NOMBRE)
FROM   ( SELECT COUNT(*) AS NOMBRE
         FROM S_NEWS.T_NEWS_NEW
         GROUP BY FRM_ID)
```

Une fois fait, il ne reste plus qu'à écrire une requête portant sur la table des forums liée à la table des news, et dont le comptage du nombre de news vaut le maximum calculé par la requête précédente.

EXERCICE 13 SOUS-REQUÊTES ET AGRÉGATS

Énoncé

Recherchez la news associée au fichier de plus grande taille.

Solution

Décomposition :

1) La plus grande taille de fichier :

```
SELECT MAX(FIC_TAILLE_O) AS FIC_MAX_TAILLE
FROM   S_NEWS.T_FICHIER_FIC

FIC_MAX_TAILLE
--------------------
6656
```

2) Le fichier de plus grande taille :

```
SELECT FIC_ID, NEW_ID, FIC_TAILLE_O
FROM   S_NEWS.T_FICHIER_FIC
WHERE  FIC_TAILLE_O = 6656

FIC_ID      NEW_ID      FIC_TAILLE_O
----------- ----------- ------------
6           4           6656
```

3) La news associée au fichier de plus grande taille :

```
SELECT *
FROM S_NEWS.T_NEWS_NEW
WHERE NEW_ID = 4
```

4) Imbrication avec la précédente requête :

```
SELECT *
FROM S_NEWS.T_NEWS_NEW
WHERE NEW_ID = (SELECT NEW_ID
                FROM  S_NEWS.T_FICHIER_FIC
                WHERE  FIC_TAILLE_O = 6656)
```

5) Imbrication avec la première requête :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_ID = (SELECT NEW_ID
                 FROM   S_NEWS.T_FICHIER_FIC
                 WHERE  FIC_TAILLE_O = (SELECT MAX(FIC_TAILLE_O)
                                        FROM   S_NEWS.T_FICHIER_FIC))
```

On peut aussi écrire cette requête avec une jointure des tables S_NEWS.T_NEWS_NEW et S_NEWS.T_FICHIER_FIC :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW AS NEW
       INNER JOIN S_NEWS.T_FICHIER_FIC AS FIC
             ON NEW.NEW_ID = FIC.NEW_ID
WHERE  FIC_TAILLE_O = (SELECT MAX(FIC_TAILLE_O)
                       FROM   S_NEWS.T_FICHIER_FIC)
```

Chapitre 5

EXERCICE 14 SOUS-REQUÊTES ET AGRÉGATS

Énoncé

Recherchez la news dont les fichiers associés occupent le plus fort volume.

Solution

Décomposition :

1) La somme des tailles de fichiers par news :

```
SELECT SUM(FIC_TAILLE_O) AS FIC_TAILLES, NEW_ID
FROM   S_NEWS.T_FICHIER_FIC
GROUP  BY NEW_ID

FIC_TAILLES          NEW_ID
-------------------- -----------
12288                2
7680                 4
1024                 10
2048                 17
```

2) Recherche du « maximum » parmi ces résultats :

```
SELECT MAX(FIC_TAILLES) AS FIC_TAILLES_MAX
FROM   (SELECT SUM(FIC_TAILLE_O) AS FIC_TAILLES, NEW_ID
        FROM   S_NEWS.T_FICHIER_FIC
        GROUP  BY NEW_ID)

FIC_TAILLES_MAX
--------------------
12288
```

3) La news dont la somme des tailles de fichiers associés vaut 12 288.

```
SELECT SUM(FIC_TAILLE_O) AS FIC_TAILLES, NEW_ID
FROM   S_NEWS.T_FICHIER_FIC
GROUP  BY NEW_ID
HAVING SUM(FIC_TAILLE_O) = 12288

FIC_TAILLES          NEW_ID
-------------------- -----------
12288                2
```

4) Les détails de la news n°2 :

```
SELECT *
FROM S_NEWS.T_NEWS_NEW
WHERE NEW_ID = 2
```

5) Imbrication avec la précédente requête :

```
SELECT *
FROM S_NEWS.T_NEWS_NEW
WHERE NEW_ID = (SELECT NEW_ID
                FROM   S_NEWS.T_FICHIER_FIC
                GROUP  BY NEW_ID
                HAVING SUM(FIC_TAILLE_O) = 12288)
```

6) Imbrication complète :

```
SELECT *
FROM S_NEWS.T_NEWS_NEW
WHERE NEW_ID = (SELECT NEW_ID
                FROM  S_NEWS. S_NEWS.FICHIER_FIC
```

```
GROUP  BY NEW_ID
HAVING SUM(FIC_TAILLE_O) =
      (SELECT MAX(FIC_TAILLES) AS FIC_TAILLES_MAX
       FROM   (SELECT SUM(FIC_TAILLE_O) AS FIC_TAILLES,
                      NEW_ID
               FROM   S_NEWS.T_FICHIER_FIC
               GROUP  BY NEW_ID
              )
      )
)
```

EXERCICE 15 SOUS-REQUÊTE CORRÉLÉE

Énoncé

Quels sont les forums sur lesquels des utilisateurs ont posté des news sans y être inscrits ?

Solution

```
SELECT FRM.FRM_ID, FRM_NOM
FROM   S_NEWS.T_NEWS_NEW NEW
       INNER JOIN S_NEWS.T_FORUM_FRM FRM
             ON NEW.FRM_ID = FRM.FRM_ID
WHERE  NOT EXISTS (SELECT *
                   FROM   S_NEWS.T_UTILISATEUR_USR USR
                          INNER JOIN S_NEWS.T_J_INSCRIS_NEWS.ISC ISC
                                ON USR.USR_ID = ISC.USR_ID
                   WHERE  NEW.USR_ID = USR.USR_ID)
```

La première requête trouve les news liées aux forums. La seconde fournit les utilisateurs inscrits aux forums. Dans le NOT EXISTS, la corrélation se fait entre les deux requêtes sur l'identifiant de l'utilisateur.

EXERCICE 16 SOUS-REQUÊTE CORRÉLÉE

Énoncé

Quel forum contient le plus fort volume de fichiers joints ?

Solution

```
SELECT FRM.FRM_ID, FRM.FRM_NOM,
       SUM(FIC.FIC_TAILLE_O) AS TAILLE_OCTET
FROM   S_NEWS.T_NEWS_NEW NEW
       LEFT OUTER JOIN S_NEWS.T_FICHIER_FIC FIC
             ON NEW.NEW_ID = FIC.NEW_ID
       INNER JOIN S_NEWS.T_FORUM_FRM FRM
             ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY FRM.FRM_ID, FRM.FRM_NOM
HAVING SUM(FIC.FIC_TAILLE_O) =
       ( SELECT MAX(TAILLE_OCTET) AS
         FROM ( SELECT SUM(FIC.FIC_TAILLE_O) AS TAILLE_OCTET, NEW.FRM_ID
                FROM   S_NEWS.T_FICHIER_FIC FIC
                       INNER JOIN S_NEWS.T_NEWS_NEW NEW
                             ON FIC.NEW_ID = NEW.NEW_ID
                GROUP  BY NEW.FRM_ID
              )
       )
```

Pour une telle requête, il convient de calculer un agrégat d'agrégat : on fait la somme des fichiers par news, puis on recherche le « maximum » dans ce résultat. Or, il n'est pas possible d'utiliser une imbrication directe d'agrégat comme MAX(SUM(...), tout simplement car les sous-ensembles que l'on doit réaliser pour ce groupage ne sont pas les mêmes. Il faut donc procéder avec deux requêtes imbriquées. La première calcule la somme des tailles de fichiers par news :

```
SELECT SUM(FIC.FIC_TAILLE_O) AS TAILLE_OCTET, NEW.FRM_ID
               FROM   S_NEWS.T_FICHIER_FIC FIC
                      INNER JOIN S_NEWS.T_NEWS_NEW NEW
                            ON FIC.NEW_ID = NEW.NEW_ID
               GROUP  BY NEW.FRM_ID
```

La seconde extrait le « maximum » à partir du résultat de la précédente :

```
SELECT MAX(TAILLE_OCTET) AS
        FROM ( SELECT SUM(FIC.FIC_TAILLE_O) AS TAILLE_OCTET, NEW.FRM_ID
               FROM   S_NEWS.T_FICHIER_FIC FIC
                      INNER JOIN S_NEWS.T_NEWS_NEW NEW
                            ON FIC.NEW_ID = NEW.NEW_ID
               GROUP  BY NEW.FRM_ID
             )
```

Il faut ensuite imbriquer ce « maximum » dans une requête semblable à la première, en le filtrant à l'aide de la clause HAVING :

```
SELECT FRM.FRM_ID, FRM.FRM_NOM,
       SUM(FIC.FIC_TAILLE_O) AS TAILLE_OCTET
FROM   S_NEWS.T_NEWS_NEW NEW
       LEFT OUTER JOIN S_NEWS.T_FICHIER_FIC FIC
             ON NEW.NEW_ID = FIC.NEW_ID
       INNER JOIN S_NEWS.T_FORUM_FRM FRM
             ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY FRM.FRM_ID, FRM.FRM_NOM
HAVING SUM(FIC.FIC_TAILLE_O) = ...
```

EXERCICE 17 SOUS-REQUÊTE CORRÉLÉE

Énoncé

Quels sont les utilisateurs qui ont posté le plus grand nombre de news dans chaque forum ?

Solution

```
SELECT FRM.FRM_ID, FRM.FRM_NOM, COUNT(NEW_ID) AS NEW_NOMBRE,
       USR.USR_ID, USR.USR_NOM
FROM   S_NEWS.T_NEWS_NEW NEW
       LEFT OUTER JOIN S_NEWS.T_UTILISATEUR_USR USR
            ON NEW.USR_ID = USR.USR_ID
       LEFT OUTER JOIN  S_NEWS.T_FORUM_FRM FRM
            ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY FRM.FRM_ID, NEW.FRM_ID, FRM.FRM_NOM, USR.USR_ID,
          NEW.USR_ID, USR.USR_NOM
```

```
HAVING COUNT(NEW_ID) =
   (SELECT MAX(NEW_NOMBRE)
    FROM   (SELECT FM2.FRM_ID, COUNT(NEW_ID) AS NEW_NOMBRE
            FROM   S_NEWS.T_NEWS_NEW NW2
                   LEFT OUTER JOIN S_NEWS.T_UTILISATEUR_USR US2
                        ON NW2.USR_ID = US2.USR_ID
                   LEFT OUTER JOIN S_NEWS. S_NEWS.FORUM_FRM FM2
                        ON NW2.FRM_ID = FM2.FRM_ID
                   WHERE  FM2.FRM_ID = FRM.FRM_ID
                   GROUP  BY FM2.FRM_ID, US2.USR_ID )
            GROUP BY FRM_ID)
```

Ici, il faut compter le nombre de news par utilisateurs et par forums, puis trouver le « maximum » par forums. Ensuite, il faut reporter ce résultat dans le filtre sur agrégat (HAVING) de la requête externe en corrélant la sous-requête et la requête principale sur le forum et l'utilisateur.

EXERCICE 18 SOUS-REQUÊTE CORRÉLÉE

Énoncé

Recherchez la question la plus récente dans chaque forum de news.

Solution

Solution avec EXISTS :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW NEW
WHERE  EXISTS (SELECT *
               FROM   S_NEWS.T_NEWS_NEW
               WHERE  NEW_ID_PERE IS NULL
                 AND  FRM_ID = NEW.FRM_ID
               GROUP  BY FRM_ID
               HAVING NEW.NEW_MOMENT = MAX(NEW_MOMENT))
```

Solution avec égalité :

```
SELECT *
FROM   S_NEWS.T_NEWS_NEW NEW
WHERE  NEW_MOMENT = (SELECT MAX(NEW_MOMENT)
                     FROM   S_NEWS.T_NEWS_NEW
                     WHERE  NEW_ID_PERE IS NULL
                       AND  FRM_ID = NEW.FRM_ID
                     GROUP  BY FRM_ID)
```

Dans ces deux solutions, le principe est de corréler la sous-requête sur le forum, de la filtrer sur l'absence de NEW_PERE_ID, et de faire correspondre le moment auquel la news a été postée au « maximum » de tous les moments pour toutes les news regroupées par forums.

Chapitre 5

EXERCICE 19 SOUS-REQUÊTES CORRÉLÉES

Énoncé

Soit un club de jeu d'échecs composé des tables suivantes :

```
S_NEWS.JOUEUR_JOR :
JOR_ID      JOR_NOM
----------- ---------------
1           Paul
2           Marcel
3           Alain
4           Jacques
5           Eric
6           Gérard

S_NEWS.PARTIE_PTI :

PTI_ID      PTI_DATEHEURE
----------- -----------------------
100         2004-11-11 08:30:00.000
101         2004-11-11 11:00:00.000
102         2004-11-11 15:00:00.000
103         2004-11-11 16:30:00.000
104         2004-11-11 19:00:00.000

TJ_JOUE_JOE :
PTI_ID      JOR_ID
----------- -----------
100         1
100         2
101         3
101         4
102         2
103         5
```

1) Quels joueurs peuvent participer à une partie ?

2) Quels joueurs peuvent compléter une partie ?

Pour ces deux requêtes, affichez les colonnes JOR_ID, JOR_NOM, PTI_ID et PTI_DATEHEURE.

Solution

1) Joueurs pouvant participer à une partie :

```
SELECT JOR_ID, JOR_NOM, PTI_ID, PTI_DATEHEURE
FROM   S_NEWS.JOUEUR_JOR JOR
       CROSS JOIN S_NEWS.PARTIE_PTI PTI
WHERE  NOT EXISTS (SELECT *
                   FROM   TJ_JOUE_JOE
                   WHERE  JOR_ID = JOR.JOR_ID
                     AND  PTI_ID = PTI.PTI_ID)
  AND  NOT EXISTS (SELECT *
                   FROM   TJ_JOUE_JOE
                   WHERE  PTI_ID = PTI.PTI_ID
                   GROUP  BY PTI_ID
                   HAVING COUNT(*) >= 2)
```

Pour participer à une partie, il faut tenir compte des paramètres suivants :

- Un joueur potentiel ne doit pas être déjà inscrit, car il ne peut jouer contre lui-même :

```
NOT EXISTS (SELECT *
            FROM   TJ_JOUE_JOE
            WHERE  JOR_ID = JOR.JOR_ID
              AND  PTI_ID = PTI.PTI_ID)
```

- Deux joueurs au minimum doivent déjà être inscrits à la partie :

```
NOT EXISTS (SELECT *
            FROM   TJ_JOUE_JOE
            WHERE  PTI_ID = PTI.PTI_ID
            GROUP  BY PTI_ID
            HAVING COUNT(*) >= 2)
```

2) Joueurs pouvant compléter une partie :

```
SELECT JOR_ID, JOR_NOM, PTI_ID, PTI_DATEHEURE
FROM   S_NEWS.JOUEUR_JOR JOE
       CROSS JOIN S_NEWS.PARTIE_PTI PTI
WHERE  NOT EXISTS (SELECT *
                   FROM   TJ_JOUE_JOE
                   WHERE  JOR_ID = JOE.JOR_ID
                     AND  PTI_ID = PTI.PTI_ID)
  AND  EXISTS (SELECT PTI_ID
               FROM   TJ_JOUE_JOE
               WHERE  PTI_ID = PTI.PTI_ID
               GROUP  BY PTI_ID
               HAVING COUNT(*) = 1)
ORDER BY JOR_ID, PTI_ID
```

Pour compléter une partie, il faut tenir compte des paramètres suivants :

- Un joueur potentiel ne doit pas être déjà inscrit, car il ne peut jouer contre lui-même :

```
NOT EXISTS (SELECT *
            FROM   TJ_JOUE_JOE
            WHERE  JOR_ID = JOE.JOR_ID
              AND  PTI_ID = PTI.PTI_ID)
```

- Il faut qu'il y ait déjà un joueur inscrit (pour faire le second) :

```
EXISTS (SELECT PTI_ID
        FROM   TJ_JOUE_JOE
        WHERE  PTI_ID = PTI.PTI_ID
        GROUP  BY PTI_ID
        HAVING COUNT(*) = 1)
```

EXERCICE 20 CLASSEMENT

Énoncé

Déterminez les trois utilisateurs ayant posté le plus grand nombre de news, avec leur rang (1, 2, 3) et le nombre de news postées. Pour cet exercice, vous ne devez pas faire appel aux fonctions analytiques ni à la clause `WINDOW`.

Solution

```
SELECT  COUNT(*) AS RANG, USR_NOM, T1.NEW_NOMBRE
FROM   (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
```

```
        FROM   S_NEWS.T_NEWS_NEW NW1
        GROUP  BY NW1.USR_ID) T1
       INNER JOIN (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
                   FROM   S_NEWS.T_NEWS_NEW NW1
                   GROUP  BY NW1.USR_ID) T2
             ON T1.NEW_NOMBRE <= T2.NEW_NOMBRE
       INNER JOIN S_NEWS.T_UTILISATEUR_USR USR
             ON T1.USR_ID = USR.USR_ID
GROUP BY T1.NEW_NOMBRE, USR_NOM
HAVING COUNT(*) <= 3
ORDER BY 1
```

En règle générale, pour effectuer un comptage, il convient de réaliser une inéquijointure de la table sur elle-même, associée à un comptage. Ici, la difficulté vient du fait que les tables devant être jointes de la sorte sont des sous-requêtes.

On peut aussi exprimer cette requête avec une expression de table :

```
WITH T AS (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
           FROM   S_NEWS.T_NEWS_NEW NW1
           GROUP  BY NW1.USR_ID)
SELECT  COUNT(*) AS RANG, USR_NOM, T1.NEW_NOMBRE
FROM   T T1
       INNER JOIN T T2
             ON T1.NEW_NOMBRE <= T2.NEW_NOMBRE
       INNER JOIN S_NEWS.T_UTILISATEUR_USR USR
             ON T1.USR_ID = USR.USR_ID
GROUP BY T1.NEW_NOMBRE, USR_NOM
HAVING COUNT(*) <= 3
ORDER BY 1
```

Une autre problématique peut se poser s'il existe des *ex aequo* dans les rangs limites (ici troisième). Dans ce cas, il faut calculer la valeur limite du nombre de news en utilisant une sous-requête. Prenons, par exemple, le classement des quatre premiers utilisateurs ayant posté le plus de news. La requête pour les extraire est la suivante :

```
SELECT  COUNT(*) AS RANG, USR_NOM, TX1.NEW_NOMBRE
FROM   (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
        FROM   S_NEWS.T_NEWS_NEW NW1
        GROUP  BY NW1.USR_ID) TX1
       INNER JOIN (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
                   FROM   S_NEWS.T_NEWS_NEW NW1
                   GROUP  BY NW1.USR_ID) TX2
             ON TX1.NEW_NOMBRE <= TX2.NEW_NOMBRE
       INNER JOIN S_NEWS.T_UTILISATEUR_USR USR
             ON TX1.USR_ID = USR.USR_ID
WHERE TX1.NEW_NOMBRE >=
(SELECT  DISTINCT TT1.NEW_NOMBRE
 FROM
 (SELECT COUNT(*) AS RANG, USR_NOM, T1.NEW_NOMBRE
  FROM   (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
          FROM   S_NEWS.T_NEWS_NEW NW1
          GROUP  BY NW1.USR_ID) T1
         INNER JOIN (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
                     FROM   S_NEWS.T_NEWS_NEW NW1
                     GROUP  BY NW1.USR_ID) T2
               ON T1.NEW_NOMBRE <= T2.NEW_NOMBRE
         INNER JOIN S_NEWS.T_UTILISATEUR_USR USR
               ON T1.USR_ID = USR.USR_ID
  GROUP BY T1.NEW_NOMBRE, USR_NOM) TT1
 INNER JOIN
```

```
 (SELECT COUNT(*) AS RANG, USR_NOM, T1.NEW_NOMBRE
  FROM   (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
          FROM   S_NEWS.T_NEWS_NEW NW1
          GROUP  BY NW1.USR_ID) T1
         INNER JOIN (SELECT NW1.USR_ID, COUNT(NW1.NEW_ID) AS NEW_NOMBRE
                     FROM   S_NEWS.T_NEWS_NEW NW1
                     GROUP  BY NW1.USR_ID) T2
               ON T1.NEW_NOMBRE <= T2.NEW_NOMBRE
         INNER JOIN S_NEWS.T_UTILISATEUR_USR USR
               ON T1.USR_ID = USR.USR_ID
  GROUP BY T1.NEW_NOMBRE, USR_NOM) TT2
 ON TT1.RANG >= TT2.RANG
 GROUP BY TT1.NEW_NOMBRE
 HAVING COUNT(DISTINCT TT2.NEW_NOMBRE) = 4)
GROUP BY TX1.NEW_NOMBRE, USR_NOM
ORDER BY 1
```

Dans la sous-requête introduite par WHERE TX1.NEW_NOMBRE >=, on trouve la même structure que vue précédemment, mais cette fois-ci elle sert à déterminer la valeur de la limite, et non le rang.

EXERCICE 21 CLASSEMENT

Énoncé

Affichez les utilisateurs avec leur rang (1, 2, 3, etc.) dans l'ordre alphabétique des noms et prénoms. Pour cet exercice, vous ne devez pas faire appel aux fonctions analytiques ni à la clause `WINDOW`.

Solution

```
SELECT USR_NOM, USR_PRENOM, RANG
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN
          (SELECT T1.USR_ID, COUNT(*) AS RANG
           FROM   (SELECT USR_ID, USR_NOM ||
                          CAST(COALESCE(USR_PRENOM, '') AS CHAR(32)) ||
                          CAST(USR_ID AS VARCHAR(16)) AS NOM
                   FROM   S_NEWS.T_UTILISATEUR_USR) T1
              INNER JOIN
                  (SELECT USR_ID, USR_NOM ||
                          CAST(COALESCE(USR_PRENOM, '') AS CHAR(32)) ||
                          CAST(USR_ID AS VARCHAR(16)) AS NOM
                   FROM   S_NEWS.UTILISATEUR_USR) T2
              ON T1.NOM >= T2.NOM
          GROUP BY T1.USR_ID ) T3
       ON USR.USR_ID = T3.USR_ID
ORDER BY 3
```

Encore une fois, c'est une inéquijointure avec comptage qu'il faut utiliser. Dans ce cas, il convient de concaténer les noms et prénoms en tenant compte des blancs de complétude, afin que l'inégalité soit vérifiée. En effet, du fait de la suppression des blancs, des effets néfastes de concaténation pourraient entraîner un classement inexact. Par exemple, ELISE LAMOUR et LISE LAMOURE seraient considérées comme identiques ; leur concaténation sans la présence des blancs du fait du forçage en CHAR donnerait, pour ces deux cas, le même résultat : LAMOURELISE.

Chapitre 5

EXERCICE 22 STATISTIQUES

Énoncé

Donnez le nom des forums sur lesquels aucun utilisateur n'a posté plusieurs news.

Solution

```
SELECT DISTINCT FRM.FRM_ID, FRM.FRM_NOM
FROM   S_NEWS.T_NEWS_NEW NEW
       INNER JOIN S_NEWS.T_FORUM_FRM FRM
             ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY USR_ID, FRM.FRM_ID, NEW.FRM_ID, FRM.FRM_NOM
HAVING COUNT(*) = ALL (SELECT COUNT(*)
                       FROM   S_NEWS.T_NEWS_NEW
                       WHERE  NEW.FRM_ID = FRM_ID
                       GROUP BY USR_ID, FRM_ID)
```

La requête suivante :

```
SELECT FRM.FRM_ID,USR_ID, COUNT(*) NEW_NOMBRE
FROM   S_NEWS.T_NEWS_NEW NEW
       INNER JOIN S_NEWS.T_FORUM_FRM FRM
             ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY USR_ID, FRM.FRM_ID, NEW.FRM_ID, FRM.FRM_NOM
ORDER  BY 1
```

permet de trouver le nombre de news par forums et par utilisateurs.

```
FRM_ID      USR_ID      NEW_NOMBRE
----------- ----------- -----------
1           2           3
1           7           1
1           11          2
1           15          1
2           6           2
2           14          1
2           22          1
3           6           1
4           2           3
4           7           1
4           14          1
5           2           2
5           4           1
5           16          1
5           18          1
6           9           1
6           10          1
7           1           1
7           4           2
7           6           2
```

D'après les résultats produits par cette requête, les forums à retenir sont ceux dans lesquels toutes les valeurs de la colonne NEW_NOMBRE sont égales à 1. C'est le sens de la sous-requête corrélée introduite par ALL.

EXERCICE 23 STATISTIQUES

Énoncé

Combien de news sont postées en moyenne par jours et par forums ?

Solution

```
-- nombre de news par forums
SELECT COUNT(*) AS NEW_NOMBRE, FRM_ID
FROM   S_NEWS.T_NEWS_NEW
GROUP  BY FRM_ID
-- nombre de jours concernés
SELECT INTERVAL DAY CAST(MAX(NEW_MOMENT AS DATE)) - CAST(MIN(NEW_MOMENT) AS DATE) +
1
FROM S_NEWS.T_NEWS_NEW
--moyenne des news postées par jours et par forums
SELECT FRM_ID,
       CAST(COUNT(*) AS FLOAT)
              /  (SELECT  CAST(INTERVAL  DAY  CAST(MAX(NEW_MOMENT  AS  DATE))  -
CAST(MIN(NEW_MOMENT) AS DATE) + 1 AS FLOAT)
         FROM   S_NEWS.T_NEWS_NEW ) AS NEW_NOMBRE_MOYEN
FROM   S_NEWS.T_NEWS_NEW
GROUP  BY FRM_ID
```

Une erreur serait de considérer comme valable la moyenne calculée sur les seuls jours au cours desquels des news ont été postées. Ainsi, la requête :

```
SELECT FRM_ID,
       CAST(COUNT(*) AS FLOAT)
       / CAST(COUNT(CAST(NEW_MOMENT AS DATE)) AS FLOAT)
FROM   S_NEWS.T_NEWS_NEW
GROUP  BY FRM_ID
```

ne donnera pas le résultat escompté, mais la moyenne du nombre de news pour les seuls jours où des news ont été postées.

La solution pour SQL Server passe par la création d'une fonction qui expurge la partie horaire d'un type combinant la date et l'heure (le type DATE n'existe pas sous SQL Server) :

```
-- fonction remplaçant un datetime par un datetime avec heure à zero
CREATE FUNCTION S_NEWS.FN_DATETIME_AS_DATE (@DT DATETIME)
RETURNS DATETIME AS
BEGIN
   RETURN CAST(FLOOR(CAST(@DT AS FLOAT)) AS DATETIME)
END
-- nombre de requêtes par forums
SELECT COUNT(*) AS NEW_NOMBRE, FRM_ID
FROM   S_NEWS.T_NEWS_NEW
GROUP  BY FRM_ID
-- nombre de jours concernés
SELECT DATEDIFF(DAY, MIN(S_NEWS.FN_DATETIME_AS_DATE(NEW_MOMENT)),
                     MAX(S_NEWS.FN_DATETIME_AS_DATE(NEW_MOMENT))) + 1
FROM S_NEWS.T_NEWS_NEW
--moyenne des news postées par jours et par forums
SELECT FRM_ID,
       CAST(COUNT(*) AS FLOAT)
       / (SELECT CAST(DATEDIFF(DAY,
                 MIN(S_NEWS.FN_DATETIME_AS_DATE(NEW_MOMENT)),
                 MAX(S_NEWS.FN_DATETIME_AS_DATE(NEW_MOMENT))) + 1 AS FLOAT)
          FROM S_NEWS.T_NEWS_NEW ) AS NEW_NOMBRE_MOYEN
FROM   S_NEWS.T_NEWS_NEW
GROUP  BY FRM_ID
```

Chapitre 5

EXERCICE 24 STATISTIQUES

Énoncé

Soit la table S_NEWS.JURY_JRY contenant des films notés par un jury :

```
JRY_FILM          JRY_HITCHCOCK  JRY_LANG  JRY_KUBRICK  JRY_WELLES
--------------------------------  --------  -----------  ----------
West Side Story   17             15                     11
Dr. Jivago                                  13           11
Ben Hur           15             14        8            11
My Fair Lady      9              10        7            13
Scarface          18             18        16           17
```

1) Donnez la note moyenne décernée par chaque juré.

2) Donnez la note moyenne obtenue par chaque film.

Solution

1) La note moyenne décernée par chaque juré :

```
SELECT SUM(CAST(JRY_HITCHCOCK AS FLOAT))
       / (SELECT COUNT(JRY_HITCHCOCK)
          FROM S_NEWS.JURY_JRY) AS MOYENNE_HITCHCOCK,
       SUM(CAST(JRY_LANG AS FLOAT))
       / (SELECT COUNT(JRY_LANG)
          FROM S_NEWS.JURY_JRY) AS MOYENNE_LANG,
       SUM(CAST(JRY_KUBRICK AS FLOAT))
       / (SELECT COUNT(JRY_KUBRICK)
          FROM S_NEWS.JURY_JRY) AS MOYENNE_KUBRICK,
       SUM(CAST(JRY_WELLES AS FLOAT))
       / (SELECT COUNT(JRY_WELLES)
          FROM S_NEWS.JURY_JRY) AS MOYENNE_WELLES
FROM  S_NEWS.JURY_JRY
```

Pensez à convertir les notes en float, sinon le calcul s'effectuera en division entière. Notez que la moyenne s'établit sur le nombre de notes attribuées par chaque juré, et non sur le nombre total de lignes.

2) La note moyenne obtenue par chaque film :

```
SELECT JRY_FILM, (CAST(COALESCE(JRY_HITCHCOCK, 0) AS FLOAT)
                  + CAST(COALESCE(JRY_LANG, 0) AS FLOAT)
                  + CAST(COALESCE(JRY_KUBRICK, 0) AS FLOAT)
                  + CAST(COALESCE(JRY_WELLES, 0) AS FLOAT) )
                  /
                 (CASE
                     WHEN JRY_HITCHCOCK IS NOT NULL THEN 1.0
                     ELSE 0.0
                   END
                  + CASE
                       WHEN JRY_LANG IS NOT NULL THEN 1.0
                       ELSE 0.0
                    END
                  + CASE
                       WHEN JRY_KUBRICK IS NOT NULL THEN 1.0
                       ELSE 0.0
```

```
                    END
                  + CASE
                      WHEN JRY_WELLES IS NOT NULL THEN 1.0
                      ELSE 0.0
                    END) AS JRY_MOYENNE
FROM  S_NEWS.JURY_JRY
```

La conclusion est que cette table mal modélisée rend l'écriture des requêtes ardue et répétitible.

EXERCICE 25 STATISTIQUES

Énoncé

En vous fondant sur la remarque formulée en conclusion de l'exercice précédent, proposez un modèle plus adapté et les requêtes associées pour traiter le problème.

Solution

Il faut découper la table en trois : une table des films, une table des jurés et une table de jointure entre les deux, avec la note de type réel :

```
CREATE TABLE S_NEWS.JURE_JUR
(JUR_ID     INTEGER NOT NULL PRIMARY KEY,
 JUR_NOM    VARCHAR(16))
CREATE TABLE S_NEWS.FILM_FLM
(FLM_ID     INTEGER NOT NULL PRIMARY KEY,
 FLM_NOM    VARCHAR(16))
CREATE TABLE TJ_NOTE_NTE
(JUR_ID     INTEGER NOT NULL,
 FLM_ID     INTEGER NOT NULL,
 NTE_NOTE   FLOAT NOT NULL,
 CONSTRAINT PK_NTE PRIMARY KEY (JUR_ID, FLM_ID),
 CONSTRAINT FK_NTE_JUR FOREIGN KEY (JUR_ID)
            REFERENCES S_NEWS.JURE_JUR (JUR_ID),
 CONSTRAINT FK_NTE_FLM FOREIGN KEY (FLM_ID)
            REFERENCES S_NEWS.FILM_FLM (FLM_ID))
```

Les requêtes s'écrivent alors :

1) La note moyenne décernée par chaque juré :

```
SELECT JUR_NOM, AVG(NTE_NOTE)
FROM   S_NEWS.JURE_JUR JUR
       INNER JOIN TJ_NOTE_NTE NTE
             ON JUR.JUR_ID = NTE.JUR_ID
GROUP BY JUR_NOM
```

2) La note moyenne obtenue par chaque film :

```
SELECT FLM_NOM, AVG(NTE_NOTE)
FROM   S_NEWS.FILM_FLM FLM
       INNER JOIN TJ_NOTE_NTE NTE
             ON FLM.FLM_ID = NTE.FLM_ID
GROUP BY FLM_NOM
```

Chapitre 5

EXERCICE 26 ENSEMBLES

Énoncé

Trouvez les utilisateurs qui ont posté une news dans un forum qui n'est pas recensé par au moins un des serveurs de leur fournisseur d'accès.

Solution

```
SELECT DISTINCT USR.USR_ID, USR_NOM, USR_PRENOM, FRM_ID
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_NEWS_NEW NEW
             ON USR.USR_ID = NEW.USR_ID
EXCEPT
SELECT DISTINCT USR.USR_ID, USR_NOM, USR_PRENOM, FRM_ID
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_J_ABONNE_ABN ABN
             ON USR.USR_ID = ABN.USR_ID
             INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
                   ON ABN.FAI_ID = FAI.FAI_ID
                   INNER JOIN S_NEWS.T_SERVEUR_SRV SRV
                         ON FAI.FAI_ID = SRV.FAI_ID
                         INNER JOIN S_NEWS.T_J_RECENSE_RCS RCS
                               ON SRV.SRV_ID = RCS.SRV_ID
```

Si votre SGBDR n'implémente pas l'opérateur SQL EXCEPT de différence, vous pouvez utiliser une construction à l'aide de NOT EXISTS :

```
SELECT DISTINCT UR1.USR_ID, USR_NOM, USR_PRENOM, FRM_ID
FROM   S_NEWS.T_UTILISATEUR_USR UR1
       INNER JOIN S_NEWS.T_NEWS_NEW NW1
             ON UR1.USR_ID = NW1.USR_ID
WHERE NOT EXISTS (
SELECT *
FROM   S_NEWS.T_UTILISATEUR_USR USR
       INNER JOIN S_NEWS.T_J_ABONNE_ABN ABN
             ON USR.USR_ID = ABN.USR_ID
             INNER JOIN S_NEWS.T_FOURNINES_NEWS.FAI FAI
                   ON ABN.FAI_ID = FAI.FAI_ID
                   INNER JOIN S_NEWS.T_SERVEUR_SRV SRV
                         ON FAI.FAI_ID = SRV.FAI_ID
                         INNER JOIN S_NEWS.T_J_RECENSE_RCS RCS
                               ON SRV.SRV_ID = RCS.SRV_ID
WHERE  USR.USR_ID = UR1.USR_ID
  AND  RCS.FRM_ID = NW1.FRM_ID)
```

EXERCICE 27 SOUS-REQUÊTES CORRÉLÉES

Énoncé

Reprenez les tables de l'exercice 19 et modifiez la table S_NEWS.PARTIE_PTI comme suit :

```
PTI_ID  PTI_DATEHEURE            PTI_DATEHEURE_FIN
------- ----------------------- -----------------------
100     2004-11-11 08:30:00.000  2004-11-11 11:30:00.000
101     2004-11-11 11:00:00.000  2004-11-11 14:00:00.000
102     2004-11-11 15:00:00.000  2004-11-11 18:00:00.000
103     2004-11-11 16:30:00.000  2004-11-11 19:30:00.000
104     2004-11-11 19:00:00.000  2004-11-11 22:00:00.000
```

Reprenez les demandes formulées :

1) Quels joueurs peuvent participer à une partie ?

2) Quels joueurs peuvent compléter une partie ?

Sachant qu'un joueur n'est pas libre tant qu'une partie n'est pas terminée, quelle condition faut-il ajouter pour tenir compte de ces nouvelles contraintes ?

Solution

Il faut que le moment de début de la partie soit supérieur à celui de la fin de toutes les parties antérieures auxquelles le joueur considéré participe. Cela peut se formuler comme suit :

```
AND PTI.PTI_DATEHEURE >= ALL (SELECT PTI_DATEHEURE_FIN
                              FROM   S_NEWS.PARTIE_PTI PT1
                                     INNER JOIN TJ_JOUE_JOE JE1
                                           ON PT1.PTI_ID = JE1.PTI_ID
                              WHERE  PTI_DATEHEURE < PTI.PTI_DATEHEURE
                                AND  JOR.JOR_ID = JE1.JOR_ID)
```

EXERCICE 28 SOUS-REQUÊTES CORRÉLÉES

Énoncé

Un site web enregistre le cours des actions sur des durées plus ou moins brèves, en provenance de différentes sources qui peuvent être contradictoires. Les données sont les suivantes :

```
CRS_ID  CRS_ACTION   CRS_VALEUR CRS_DEBUT         CRS_FIN
------- ------------ ---------- ----------------- ----------------
1       IBM             133.20  2005-01-01 09:00  2005-01-01 10:22
2       IBM             133.20  2005-01-01 11:33  2005-01-01 16:15
3       Air France     1214.21  2005-01-01 10:00  2005-01-01 12:00
4       Air France     1215.44  2005-01-01 12:00  2005-01-01 13:00
5       IBM             135.10  2005-01-01 16:00  2005-01-01 17:00
6       Air France     1215.44  2005-01-01 12:45  2005-01-01 14:30
7       Air France     1215.44  2005-01-01 13:00  2005-01-01 14:30
8       Air France     1225.11  2005-01-01 14:15  2005-01-01 14:50
9       Air France     1225.11  2005-01-01 14:50  2005-01-01 18:30
10      Air France     1215.44  2005-01-01 13:15  2005-01-01 14:00
```

Certaines périodes se recoupent avec les mêmes informations.

Énoncé *(suite)*

Mettez au point une requête capable de proposer les durées les plus pertinentes pendant lesquelles le cours des actions a été le même, afin de minimiser les lignes extraites.

Votre requête doit donner la solution suivante :

```
CRS_ACTION  CRS_DEBUT         CRS_FIN           CRS_VALEUR
----------- ----------------- ----------------- ----------
Air France  2005-01-01 10:00  2005-01-01 12:00     1214.21
Air France  2005-01-01 12:00  2005-01-01 14:30     1215.44
Air France  2005-01-01 14:15  2005-01-01 18:30     1225.11
IBM         2005-01-01 09:00  2005-01-01 10:22      133.20
IBM         2005-01-01 11:33  2005-01-01 16:15      133.20
IBM         2005-01-01 16:00  2005-01-01 17:00      135.10
```

Solution

On peut considérer toutes les relations de la table sur elle-même par actions, sans tenir compte des aspects temporels. On éliminera ensuite les périodes pendant lesquelles un « trou » d'intervalle apparaît, ainsi que les périodes plus petites incluses dans une période plus grande.

```
SELECT DISTINCT CR1.CRS_ACTION, CR1.CRS_DEBUT, CR2.CRS_FIN, CR1.CRS_VALEUR
FROM   S_NEWS.COURS_CRS CR1
       INNER JOIN S_NEWS.COURS_CRS CR2
             ON      CR1.CRS_ACTION = CR2.CRS_ACTION
                AND  CR1.CRS_VALEUR = CR2.CRS_VALEUR
                AND  CR1.CRS_DEBUT < CR2.CRS_FIN

-- élimination des trous entre CR1.CRS_DEBUT et CR2.CRS_FIN
  AND NOT EXISTS
      (SELECT *
       FROM   S_NEWS.COURS_CRS CR3
       WHERE  CR3.CRS_ACTION = CR1.CRS_ACTION
         AND  CR3.CRS_VALEUR = CR1.CRS_VALEUR
         AND  CR3.CRS_DEBUT  > CR1.CRS_FIN
         AND  CR3.CRS_DEBUT  <= CR2.CRS_DEBUT
         AND  NOT EXISTS
              (SELECT *
               FROM   S_NEWS.COURS_CRS CR4
               WHERE  CR4.CRS_ACTION =  CR1.CRS_ACTION
                 AND  CR4.CRS_VALEUR =  CR1.CRS_VALEUR
                 AND  CR4.CRS_DEBUT  <  CR3.CRS_DEBUT
                 AND  CR4.CRS_FIN    >= CR3.CRS_DEBUT) )
-- élimination des sous périodes
  AND NOT EXISTS
      (SELECT *
       FROM   S_NEWS.COURS_CRS CR5
       WHERE  CR5.CRS_ACTION =  CR1.CRS_ACTION
         AND  CR5.CRS_VALEUR =  CR1.CRS_VALEUR
         AND  (   (CR5.CRS_DEBUT <  CR1.CRS_DEBUT
                        AND CR5.CRS_FIN   >= CR1.CRS_DEBUT)
               OR (CR5.CRS_DEBUT <= CR2.CRS_FIN
                        AND CR5.CRS_FIN >  CR2.CRS_FIN  ) ) )
ORDER BY CR1.CRS_ACTION, CR1.CRS_DEBUT
```

Voici une autre façon de procéder pour résoudre ce problème.

```
SELECT CRS_ACTION, CRS_DEBUT, MIN(CRS_FIN), CRS_VALEUR
FROM
(SELECT DISTINCT CR1.CRS_ACTION, CR1.CRS_DEBUT, CR2.CRS_FIN,
        CR1.CRS_VALEUR
 FROM   S_NEWS.COURS_CRS CR1
        INNER JOIN S_NEWS.COURS_CRS CR2
              ON     CR1.CRS_FIN    <= CR2.CRS_FIN
                 AND CR1.CRS_ACTION =  CR2.CRS_ACTION
                 AND CR1.CRS_VALEUR =  CR2.CRS_VALEUR
        INNER JOIN S_NEWS.COURS_CRS CR3
              ON     CR1.CRS_ACTION =  CR3.CRS_ACTION
                 AND CR1.CRS_VALEUR =  CR3.CRS_VALEUR
 GROUP BY CR1.CRS_ACTION, CR1.CRS_DEBUT, CR2.CRS_FIN, CR1.CRS_VALEUR
 HAVING COUNT(CASE
                 WHEN (    CR3.CRS_DEBUT <  CR1.CRS_DEBUT
                       AND CR1.CRS_DEBUT <= CR3.CRS_FIN  )
                   OR (    CR3.CRS_DEBUT <= CR2.CRS_FIN
                       AND CR2.CRS_FIN   <  CR3.CRS_FIN  )
                 THEN 1
               END) = 0 )
GROUP BY CRS_ACTION, CRS_DEBUT,CRS_VALEUR
```

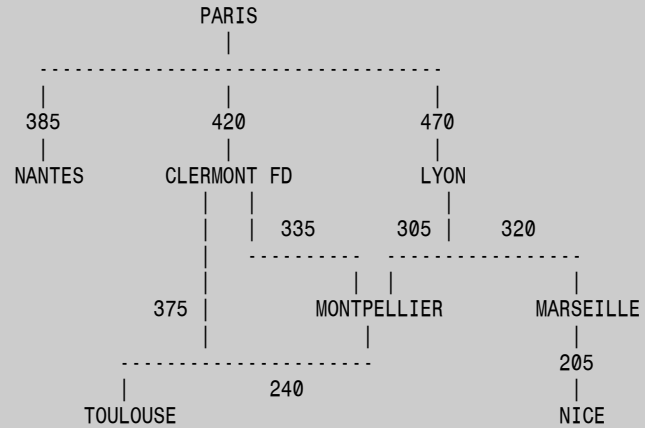
EXERCICE 29 GRAPHE

Énoncé

Soit la table des trajets S_NEWS.TRAJES_NEWS.TRJ suivante :

TRJ_VILLE_DEPART	TRJ_VILLE_ARRIVEE	TRJ_KILOMETRAGE
PARIS	NANTES	385
PARIS	CLERMONT-FERRAND	420
PARIS	LYON	470
CLERMONT-FERRAND	MONTPELLIER	335
CLERMONT-FERRAND	TOULOUSE	375
LYON	MONTPELLIER	305
LYON	MARSEILLE	320
MONTPELLIER	TOULOUSE	240
MARSEILLE	NICE	205

représentant un maillage de transport :



Énoncé *(suite)*

1) Donnez la liste de tous les trajets avec le nombre d'étapes depuis Paris.

2) Donnez la liste de tous les trajets depuis Paris qui comprennent au moins deux étapes.

3) Donnez la liste de tous les trajets depuis Paris, avec la distance en kilomètres.

4) Donnez la distance la plus courte pour le trajet Paris-Toulouse.

Solution

1) Liste de tous les trajets avec le nombre d'étapes depuis Paris :

```
WITH RECURSIVE trajet (VILLE_ARRIVEE, NB_ETAPE)
AS
   (SELECT DISTINCT TRJ_VILLE_DEPART AS VILLE_ARRIVEE, 0 AS NB_ETAPE
    FROM   S_NEWS.TRAJES_NEWS.TRJ
    WHERE  TRJ_VILLE_DEPART = 'PARIS'
    UNION  ALL
    SELECT TRJ_VILLE_ARRIVEE, depart.NB_ETAPE + 1
    FROM   S_NEWS.TRAJES_NEWS.TRJ AS arrivee
           INNER JOIN trajet AS depart
                 ON depart.VILLE_ARRIVEE = arrivee.TRJ_VILLE_DEPART)
SELECT VILLE_ARRIVEE, NB_ETAPE
FROM   trajet
```

La solution passe par une requête récursive bâtie à l'aide d'une expression de table :

```
VILLE_ARRIVEE       NB_ETAPE
------------------- -----------------
PARIS               0
NANTES              1
CLERMONT-FD         1
LYON                1
MONTPELLIER         2
MONTPELLIER         2
MARSEILLE           2
TOULOUSE            2
TOULOUSE            3
TOULOUSE            3
NICE                3
```

Voici les différentes passes par lesquelles SQL arrive à fournir le jeu de résultats précédent :

```
VILLE_DEPART  VILLE_AR_PASSE_1  VILLE_AR_PASSE_2  VILLE_AR_PASSE_3
------------  ----------------  ----------------  ----------------
PARIS
PARIS         NANTES
PARIS         CLERMONT-FD
PARIS         LYON
PARIS         CLERMONT-FD       MONTPELLIER
PARIS         LYON              MONTPELLIER
PARIS         LYON              MARSEILLE
PARIS         CLERMONT-FD       TOULOUSE
PARIS         CLERMONT-FD       MONTPELLIER       TOULOUSE
PARIS         LYON              MONTPELLIER       TOULOUSE
PARIS         LYON              MARSEILLE         NICE
```

Pour obtenir le niveau, on commence par fixer la valeur zéro pour l'étape de départ puis on incrémente d'une unité à chaque nouvelle passe.

Solution avec Oracle :

```
SELECT TRJ_VILLE_ARRIVEE AS VILLE_ARRIVEE, LEVEL AS NB_ETAPE
FROM   S_NEWS.TRAJES_NEWS.TRJ
START  WITH TRJ_VILLE_DEPART = 'Paris'
CONNECT BY PRIOR TRJ_VILLE_ARRIVEE = TRJ_VILLE_DEPART
```

Oracle utilise une syntaxe particulière pour les parcours récursifs. START WITH indique l'étape de départ, CONNECT BY stipule la manière dont est liée la hiérarchie, et le mot clé PRIOR indique le sens de parcours. La pseudo colonne LEVEL réalise automatiquement le comptage des différentes passes.

2) Liste de tous les trajets depuis Paris, qui contiennent au moins deux étapes :

```
WITH RECURSIVE trajet (VILLE_ARRIVEE, NB_ETAPE)
AS
   (SELECT DISTINCT TRJ_VILLE_DEPART AS VILLE_ARRIVEE, 0 AS NB_ETAPE
    FROM   S_NEWS.TRAJES_NEWS.TRJ
    WHERE  TRJ_VILLE_DEPART = 'PARIS'
    UNION  ALL
    SELECT TRJ_VILLE_ARRIVEE, depart.NB_ETAPE + 1
    FROM   S_NEWS.TRAJES_NEWS.TRJ AS arrivee
           INNER JOIN trajet AS depart
                 ON depart.VILLE_ARRIVEE = arrivee.TRJ_VILLE_DEPART)
SELECT VILLE_ARRIVEE, NB_ETAPE
FROM   trajet
WHERE  NB_ETAPE >= 2
VILLE_ARRIVEE       NB_ETAPE
------------------- -----------------
MONTPELLIER         2
MONTPELLIER         2
MARSEILLE           2
TOULOUSE            2
TOULOUSE            3
TOULOUSE            3
NICE                3
```

Il suffit d'ajouter à la première requête un filtre WHERE dans l'expression de requête.

3) Liste de tous les trajets depuis Paris, avec la distance en kilomètres :

```
WITH RECURSIVE trajet (VILLE_ARRIVEE, DISTANCE)
AS
   (SELECT DISTINCT TRJ_VILLE_DEPART AS VILLE_ARRIVEE, 0 AS DISTANCE
    FROM   S_NEWS.TRAJES_NEWS.TRJ
    WHERE  TRJ_VILLE_DEPART = 'PARIS'
    UNION  ALL
    SELECT TRJ_VILLE_ARRIVEE, depart.DISTANCE + arrivee.TRJ_KILOMETRAGE
    FROM   S_NEWS.TRAJES_NEWS.TRJ AS arrivee
           INNER JOIN trajet AS depart
                 ON depart.VILLE_ARRIVEE = arrivee.TRJ_VILLE_DEPART)

SELECT VILLE_ARRIVEE, DISTANCE
FROM   trajet
```

```
VILLE_ARRIVEE       NB_ETAPE
------------------- -----------------
PARIS               0
NANTES              385
CLERMONT-FERRAND    420
LYON                470
MONTPELLIER         755
MONTPELLIER         775
MARSEILLE           790
TOULOUSE            795
TOULOUSE            995
TOULOUSE            1015
NICE                995
```

Comme pour le niveau, pour calculer la distance on commence par fixer la valeur zéro pour l'étape de départ, puis on incrémente de la distance entre les villes à chaque nouvelle passe.

4) Calcul de la distance la plus courte pour le trajet Paris-Toulouse :

```
WITH RECURSIVE trajet (VILLE_ARRIVEE, DISTANCE)
AS
   (SELECT DISTINCT TRJ_VILLE_DEPART AS VILLE_ARRIVEE, 0 AS DISTANCE
    FROM   S_NEWS.TRAJES_NEWS.TRJ
    WHERE  TRJ_VILLE_DEPART = 'PARIS'
    UNION  ALL
    SELECT TRJ_VILLE_ARRIVEE, depart.DISTANCE + arrivee.TRJ_KILOMETRAGE
    FROM   S_NEWS.TRAJES_NEWS.TRJ AS arrivee
           INNER JOIN trajet AS depart
                 ON depart.VILLE_ARRIVEE = arrivee.TRJ_VILLE_DEPART)
SELECT VILLE_ARRIVEE, MIN(DISTANCE) AS DISTANCE
FROM   trajet
WHERE  VILLE_ARRIVEE = 'TOULOUSE'
GROUP  BY VILLE_ARRIVEE
```

En mathématique comme en algorithmique, ce problème de graphe est connu sous le nom de « problème du voyageur de commerce ».

EXERCICE 30 LIMITE

Énoncé

Quels sont les six premiers utilisateurs dans l'ordre alphabétique des prénoms ? Pour résoudre cet exercice, vous devez utiliser une fonction analytique et le concept de fenêtrage.

Cet exercice a été proposé au chapitre précédent (exercice 18) et se révélait sans solution à ce stade malgré sa simplicité.

Solution

En toute logique, les solutions valides doivent être construites par encapsulation de la requête qui construit le rang en tant que sous-requête d'une requête mère filtrant ce rang.

Voici les deux solutions, suivant que l'on considère le rang ou l'énumération des lignes :

```
SELECT *
FROM   (SELECT DISTINCT USR_NOM, USR_PRENOM,
               ROW_NUMBER() OVER(ORDER BY USR_PRENOM) AS RANG
        FROM   S_NEWS.T_UTILISATEUR_USR
        WHERE  USR_PRENOM IS NOT NULL)

WHERE RANG <= 5
USR_NOM           USR_PRENOM     RANG
----------------- ---------------------
DUPONT            Charles        4
DUPONT            Charles        5
ENTE              Aurélia        2
MAILLANT          Catherine      3
VALLADON          Antoine        1
SELECT *
FROM   (SELECT DISTINCT USR_NOM, USR_PRENOM,
               DENSE_RANK() OVER(ORDER BY USR_PRENOM) AS RANG
        FROM   S_NEWS.T_UTILISATEUR_USR
        WHERE  USR_PRENOM IS NOT NULL)

WHERE RANG <= 5
USR_NOM           USR_PRENOM      RANG
----------------- --------------- -------
CLERC             François        5
DUPONT            Charles         4
DUPONT            François        5
ENTE              Aurélia         2
MAILLANT          Catherine       3
VALLADON          Antoine         1
```

EXERCICE 31 OPÉRATIONS ENSEMBLISTES

Énoncé

Soit les tables suivantes d'une entreprise de location de voitures possédant un système informatique propre à chaque site (Lyon et Toulouse) :

```
CREATE TABLE S_NEWS.VOITURE_LYON_VLN
(VLN_ID              INTEGER NOT NULL PRIMARY KEY,
 VLN_IMMATRICULATION VARCHAR(12) NOT NULL,
 VLN_MARQUE          CHAR(32),
 VLN_MODELE          CHAR(16),
 VLN_COULEUR         CHAR(16))
CREATE TABLE S_NEWS.VOITURE_TOULOUSE_VTL
(VTL_PLAQUE          CHAR(16) NOT NULL PRIMARY KEY,
 VTL_MARQUE          VARCHAR(20),
 VTL_MODELE          VARCHAR(12),
 VTL_COULEUR         VARCHAR(8),
 VTL_CARBURANT       CHAR(2))
```

Énoncé *(suite)*

```
VLN_ID   VLN_IMMATRICULATION VLN_MARQUE    VLN_MODELE    VLN_COULEUR
-------- ------------------- ------------- ------------- ------------
59       128 KLM 13          Renault       Megane        Blanc
66       458 ADR 83          Chrysler      PT Cruiser    Gris
78       147 ACB 83          Volkswagen    Touran        Gris
87       5147 XC 16          Volkswagen    Sharan        Blanc
99       845 JCV 13          Volkswagen    Touran        Blanc
69       951 FGH 16          Renault       Clio          Blanc

VTL_PLAQUE       VTL_MARQUE     VTL_MODELE   VTL_COULEUR VTL_CARBURANT
---------------- -------------- ------------ ----------- -------------
894 KDM 31       Chrysler       PT Cruiser   Gris        ES
895 KDM 31       Renault        Megane       Gris        GO
896 KDM 31       Renault        Megane       Blanc       ES
962 JKD 31       Volkswagen     Sharan       Blanc       GO
963 JKD 31       Volkswagen     Touran       Noir        GO
5841 MN 12       Peugeot        806          Vert        GO
```

1) Quels sont tous les modèles de voitures de l'entreprise ?

2) Donnez la liste de tous les véhicules de l'entreprise (immatriculation, marque, modèle, couleur).

3) Réalisez l'union des deux tables sur les colonnes modèle et couleur sans faire usage de l'opérateur ensembliste UNION.

Solution

1) Tous les modèles de voitures de l'entreprise :

```
SELECT VLN_MODELE
FROM S_NEWS.VOITURE_LYON_VLN
UNION
SELECT VTL_MODELE
FROM S_NEWS.VOITURE_TOULOUSE_VTL
```

Notez que SQL n'exige pas que les colonnes soient exactement du même type, mais d'un type compatible. Ici, même si l'on trouve VARCHAR d'un côté et CHAR de l'autre ainsi que des longueurs différentes, l'opération se réalisera au mieux.

2) La liste de tous les véhicules de l'entreprise :

```
SELECT VLN_IMMATRICULATION, VLN_MARQUE, VLN_MODELE, VLN_COULEUR
FROM   S_NEWS.VOITURE_LYON_VLN
UNION
SELECT VTL_PLAQUE, VTL_MARQUE, VTL_MODELE, VTL_COULEUR
FROM   S_NEWS.VOITURE_TOULOUSE_VTL
```

3) Union des deux tables sur les colonnes modèle et couleur, sans faire usage de l'opérateur ensembliste UNION :

```
SELECT COALESCE(VLN_MODELE, VTL_MODELE) AS MODELE,
       COALESCE(VLN_COULEUR, VTL_COULEUR) AS COULEUR
FROM   S_NEWS.VOITURE_LYON_VLN VLN
       FULL OUTER JOIN S_NEWS.VOITURE_TOULOUSE_VTL VTL
            ON VLN.VLN_MODELE = VTL.VTL_MODELE
               AND VLN.VLN_COULEUR = VTL.VTL_COULEUR
```

Pour réaliser une telle union sans recourir à l'opérateur ensembliste UNION, il suffit de faire une jointure externe bilatérale et de réunir l'une ou l'autre des valeurs NON NULL de chaque couple de valeurs formées entre les tables.

EXERCICE 32 OPÉRATIONS ENSEMBLISTES

Énoncé

Reprenez les tables et les données de l'exercice précédent :

1) Donnez la liste des marques communes.

2) Donnez la liste des modèles communs (avec la marque).

3) Réalisez l'intersection des deux tables sur les colonnes marque et couleur sans faire usage de l'opérateur ensembliste INTERSECT

Solution

1) Liste des marques communes :

```
SELECT VLN_MARQUE
FROM S_NEWS.VOITURE_LYON_VLN
INTERSECT
SELECT VTL_MARQUE
FROM S_NEWS.VOITURE_TOULOUSE_VTL
```

2) Liste des modèles communs (avec la marque) :

```
SELECT VLN_MODELE, VLN_MARQUE
FROM S_NEWS.VOITURE_LYON_VLN
INTERSECT
SELECT VTL_MODELE, VTL_MARQUE
FROM S_NEWS.VOITURE_TOULOUSE_VTL
```

3) Intersection des deux tables sur les colonnes marque et couleur sans faire usage de l'opérateur ensembliste INTERSECT :

```
SELECT VLN_COULEUR, VLN_MARQUE
FROM   S_NEWS.VOITURE_LYON_VLN VLN
       INNER JOIN S_NEWS.VOITURE_TOULOUSE_VTL VTL
             ON VLN.VLN_COULEUR = VTL.VTL_COULEUR
                AND VLN.VLN_MARQUE = VTL.VTL_MARQUE
```

Cette requête utilise une jointure.

```
SELECT VLN_COULEUR, VLN_MARQUE
FROM   S_NEWS.VOITURE_LYON_VLN
WHERE  (VLN_COULEUR, VLN_MARQUE) IN (SELECT VTL_COULEUR, VTL_MARQUE
                                     FROM   S_NEWS.VOITURE_TOULOUSE_VTL)
```

Cette requête utilise l'opérateur IN avec la technique du ROW VALUE CONSTRUCTOR.

```
     SELECT VLN_COULEUR, VLN_MARQUE
FROM   S_NEWS.VOITURE_LYON_VLN VLN
WHERE  EXISTS (SELECT *
               FROM   S_NEWS.VOITURE_TOULOUSE_VTL
               WHERE VLN.VLN_COULEUR = VTL_COULEUR
                 AND VLN.VLN_MARQUE =VTL_MARQUE)
```

Cette requête utilise le prédicat EXISTS.

Dans ces trois cas, notez l'utilisation du mot clé DISTINCT pour éviter les doublons. Sans cela, l'opération serait INTERSECT ALL.

EXERCICE 33 OPÉRATIONS ENSEMBLISTES

Énoncé

Reprenez les tables et les données de l'exercice 31 :

1) Donnez la liste des modèles et couleurs des véhicules du magasin de Lyon qui ne sont pas présents dans celui de Toulouse.

2) Donnez, par agences, les marques des véhicules dont le nombre diffère dans l'autre agence.

3) Réalisez la différence entre la table de Lyon et la table de Toulouse sur les colonnes marque et modèle, sans faire usage de l'opérateur ensembliste EXCEPT.

Solution

1) Liste des modèles et couleurs des véhicules du magasin de Lyon qui ne sont pas présentes dans celui de Toulouse :

```
SELECT VLN_MARQUE, VLN_COULEUR
FROM   S_NEWS.VOITURE_LYON_VLN
EXCEPT
SELECT VTL_MARQUE, VTL_COULEUR
FROM   S_NEWS.VOITURE_TOULOUSE_VTL
```

2) Marques des véhicules, par agences, dont le nombre diffère dans l'autre agence :

```
SELECT 'LYON' AS AGENCE, *
FROM   (SELECT VLN_MARQUE, COUNT(*) AS N
        FROM   S_NEWS.VOITURE_LYON_VLN
        GROUP  BY VLN_MARQUE
        EXCEPT
        SELECT VTL_MARQUE, COUNT(*) AS N
        FROM   S_NEWS.VOITURE_TOULOUSE_VTL
        GROUP  BY VTL_MARQUE)
UNION
SELECT 'TOULOUSE', *
FROM   (SELECT VTL_MARQUE, COUNT(*) AS N
        FROM   S_NEWS.VOITURE_TOULOUSE_VTL
        GROUP  BY VTL_MARQUE
        EXCEPT
        SELECT VLN_MARQUE, COUNT(*) AS N
        FROM   S_NEWS.VOITURE_LYON_VLN
        GROUP  BY VLN_MARQUE)
```

3) Différence entre la table de Lyon et la table de Toulouse sur les colonnes marque et modèle, sans faire usage de l'opérateur ensembliste EXCEPT :

```
SELECT VLN_MARQUE, VLN_MODELE
FROM   S_NEWS.VOITURE_LYON_VLN VLN
       LEFT OUTER JOIN S_NEWS.VOITURE_TOULOUSE_VTL VTL
            ON VLN.VLN_MARQUE = VTL.VTL_MARQUE
               AND VLN.VLN_MODELE = VTL.VTL_MODELE
WHERE  VTL.VTL_MARQUE IS NULL
   OR  VTL.VTL_MODELE IS NULL
```

Avec une jointure :

a) avec NOT IN et le ROW VALUE CONSTRUCTOR :

```
SELECT VLN_MARQUE, VLN_MODELE
FROM   S_NEWS.VOITURE_LYON_VLN
WHERE  (VLN_MARQUE, VLN_MODELE)
        NOT IN (SELECT VTL_MARQUE, VTL_MODELE
                FROM   S_NEWS.VOITURE_TOULOUSE_VTL VTL)
```

b) avec une sous-requête corrélée et l'opérateur NOT EXISTS :

```
SELECT VLN_MARQUE, VLN_MODELE
FROM   S_NEWS.VOITURE_LYON_VLN VLN
WHERE  NOT EXISTS (SELECT *
                   FROM   S_NEWS.VOITURE_TOULOUSE_VTL
                   WHERE  VLN.VLN_MARQUE = VTL_MARQUE
                     AND  VLN.VLN_MODELE = VTL_MODELE)
```

EXERCICE 34 EXPRESSION DE TABLE

Énoncé

Utilisez le principe de l'expression de table afin de trouver la moyenne du nombre de news postées par forums.

Solution

```
WITH
S_NEWS.NOMBRE (N, FRM) AS
   (SELECT COUNT(*), FRM_ID
    FROM   S_NEWS.T_NEWS_NEW
    GROUP  BY FRM_ID),
S_NEWS.MOYENNE (N) AS
   (SELECT AVG(N)
    FROM   S_NEWS.NOMBRE)
SELECT *
FROM S_NEWS.MOYENNE
```

SQL permet d'utiliser une expression de table bâtie à partir d'une autre expression de table définie auparavant. La première requête donne le nombre de news par forums, et la seconde calcule la moyenne fournie par la première.

Chapitre 5

EXERCICE 35 DIVISION RELATIONNELLE

Énoncé

Avec les tables suivantes décrivant des films, leurs réalisateurs et les acteurs qui jouent dans ces films :

```
-- les films :
FLM_ID    RLS_NEWS.ID   FLM_TITRE                        FLM_ANNEE FLM_PAYS
--------- -------- ------------------------------ --------- ----------
1         1        Fenêtre sur cour                1954      USA
7         2        Dr Folamour                     1963      USA
9         3        Le tigre aime la chair fraîche  1964      France
14        3        Que la bête meurt               1969      France
...
-- les réalisateurs :
RLS_NEWS.ID      RLS_NEWS.NOM
----------- --------------------
1           HITCHCOCK
2           KUBRICK
3           CHABROL
-- les acteurs :
ACS_NEWS.ID     ACS_NEWS.NOM            ACS_NEWS.PRENOM      ACS_NEWS.PAYS
----------- -------------------- --------------- ---------------
1           Hanin                Roger           France
7           Chabrol              Claude          France
9           Ronet                Maurice         France
10          Seberg               Jean            USA
16          Stewart              James           USA
17          Kelly                Grace           USA
...
-- la table de jointure entre les films et les acteurs :
FLM_ID      ACS_NEWS.ID
----------- -----------
1           16
1           17
1           18
7           27
7           28
9           1
9           2
...
```

Écrivez les requêtes permettant de trouver :

1) Les acteurs américains qui ont joué dans tous les films d'Hitchcock.

2) Les films dans lesquels les acteurs sont tous de la même nationalité.

3) Les metteurs en scène ayant produit un film chaque année.

Solution

1) Les acteurs américains qui ont joué dans tous les films d'Hitchcock.

Nous pouvons reformuler de la manière suivante : « Quels sont les acteurs tels qu'il n'existe pas un film d'Hitchcock dans lequel ils n'ont pas joué ? », et obtenir la requête ainsi :

```
SELECT *
FROM   S_NEWS.ACTEUR_ACT A
WHERE  NOT EXISTS
           (SELECT *
            FROM   S_NEWS.FILM_FLM F
```

```
            INNER JOIN S_NEWS.REALISATEUR_RLT R
                  ON F.RLS_NEWS.ID = R.RLS_NEWS.ID
      WHERE  RLS_NEWS.NOM = 'HITCHCOCK'
        AND  NOT EXISTS
                 (SELECT *
                  FROM   S_NEWS.JOUE_JOU J2
                         INNER JOIN S_NEWS.ACTEUR_ACT A2
                               ON J2.ACS_NEWS.ID = A2.ACS_NEWS.ID
                  WHERE  F.FLM_ID = J2.FLM_ID
                    AND  A.ACS_NEWS.ID = A2.ACS_NEWS.ID))
```

Il s'agit bien d'une division relationnelle.

2) Les films dans lesquels les acteurs sont tous de la même nationalité.

Là encore, reformulons la question : « Quels sont les films dans lesquels les acteurs n'ont pas une nationalité différente d'un acteur du film ? ». Nous obtenons ainsi une requête proche de la précédente :

```
SELECT *
FROM   S_NEWS.FILM_FLM F
WHERE  EXISTS
       (SELECT *
        FROM   S_NEWS.ACTEUR_ACT A2
        WHERE  NOT EXISTS
                   (SELECT *
                    FROM   S_NEWS.JOUE_JOU J3
                           INNER JOIN S_NEWS.ACTEUR_ACT A3
                                 ON J3.ACS_NEWS.ID = A3.ACS_NEWS.ID
                    WHERE  A2.ACS_NEWS.PAYS <> A3.ACS_NEWS.PAYS
                      AND  F.FLM_ID = J3.FLM_ID))
```

Il ne s'agit pas d'une division relationnelle, bien que l'écriture en soit proche. Notons par ailleurs qu'une telle requête peut aussi s'exprimer à l'aide de l'opérateur ALL :

```
SELECT *
FROM   S_NEWS.FILM_FLM F
WHERE  EXISTS
       (SELECT *
        FROM   S_NEWS.ACTEUR_ACT
        WHERE  ACS_NEWS.PAYS = ALL (SELECT ACS_NEWS.PAYS
                                    FROM   S_NEWS.JOUE_JOU J3
                                           INNER JOIN S_NEWS.ACTEUR_ACT A3
                                                 ON J3.ACS_NEWS.ID = A3.ACS_NEWS.ID
                                    WHERE  F.FLM_ID = J3.FLM_ID))
```

Cela revient à dire : « Les films dans lesquels tout acteur a la même nationalité que tous les acteurs du film ».

3) Les metteurs en scène ayant produit un film chaque année.

```
SELECT  RLS_NEWS.ID, COUNT(*)
FROM    S_NEWS.FILM_FLM F
GROUP   BY RLS_NEWS.ID
HAVING  COUNT(*) = (SELECT COUNT(DISTINCT FLM_ANNEE)
                    FROM   S_NEWS.FILM_FLM F2
                    WHERE  F.RLS_NEWS.ID = F2.RLS_NEWS.ID)
   AND  COUNT(*) = (SELECT MAX(FLM_ANNEE) - MIN(FLM_ANNEE) + 1
                    FROM   S_NEWS.FILM_FLM F3
                    WHERE  F.RLS_NEWS.ID = F3.RLS_NEWS.ID)
```

Pour exprimer cette requête, une double condition est nécessaire : le nombre de films doit être identique au nombre des différentes années au cours desquelles le réalisateur a

produit des films. C'est ce que calcule l'expression COUNT(DISTINCT FLM_ANNEE). Cela évite d'obtenir le nom des réalisateurs qui ont produit plus d'un film la même année. Mais cela ne suffit pas. Il faut aussi que le nombre de films soit égal au nombre d'années écoulées, exprimé par (MAX(FLM_ANNEE) - MIN(FLM_ANNEE) + 1).

Voici le mécanisme détaillé avec un jeu de données restreint :

```
réalisateur  années de réalisation   nb. de   années        années
                                     films    différentes   écoulées
------------ ----------------------- -------- ------------ ---------
DUPONT       1960, 1960, 1962, 1963  4        3             4
DURAND       1960, 1962, 1963, 1964  4        4             5
MARTIN       1960, 1961, 1962, 1963  4        4             4
```

Nous voyons dans ce tableau que seuls, les chiffres de MARTIN sont égaux dans les trois colonnes.

EXERCICE 36 CONCATÉNATION DE COLONNE

Énoncé

Une table de nom S_NEWS.PHRASE_PHR contient les mots d'une phrase et leur position relative. Certains mots se terminent avec une apostrophe. À l'aide d'une requête utilisant une expression de table, réalisez la recomposition de la phrase y compris en y ajoutant le point final.

```
CREATE TABLE S_NEWS.PHRASE_PHR
(PHR_ID            INTEGER NOT NULL,
 PHR_MOS_NEWS.POSITION  INTEGER NOT NULL,
 PHR_MOT           VARCHAR(32) NOT NULL,
 CONSTRAINT PK_PHR PRIMARY KEY (PHR_ID, PHR_MOS_NEWS.POSITION));

PHR_ID      PHR_MOS_NEWS.POSITION PHR_MOT
----------- --------------- --------------------------------
1           1               Le
1           2               petit
1           3               chat
1           4               est
1           5               mort
2           1               Les
2           2               sanglots
2           3               longs
2           4               des
2           5               violons
2           6               de
2           7               l'
2           8               automne
2           9               blessent
2           10              mon
2           11              coeur
2           12              d'
2           13              une
2           14              langueur
2           15              monotone
```

Solution

La solution utilise deux expressions de table enchaînées : l'une fait la concaténation mot à mot, l'autre trouve parmi les morceaux de phrases en construction celle qui possède le plus de mots. C'est à l'évidence la phrase proprement restituée.

```
WITH
phrases (phrase, id, position)
AS
(
SELECT CAST(PHR_MOT AS VARCHAR(max))
       + CASE
            WHEN SUBSTRING(PHR_MOT, LEN(PHR_MOT), 1) = '''' THEN ''
            ELSE ' '
         END, PHR_ID, PHR_MOS_NEWS.POSITION
FROM   S_NEWS.PHRASE_PHR
WHERE  PHR_MOS_NEWS.POSITION = 1
UNION ALL
SELECT phrase + CAST(PHR_MOT AS VARCHAR(max))
              + CASE
                   WHEN SUBSTRING(PHR_MOT, LEN(PHR_MOT), 1) = ''''
                      THEN ''
                   ELSE ' '
                END AS PHRASE,
       PHR_ID, PHR_MOS_NEWS.POSITION
FROM   S_NEWS.PHRASE_PHR AS suiv
       INNER JOIN phrases
             ON suiv.PHR_ID = phrases.id
                AND suiv.PHR_MOS_NEWS.POSITION = phrases.position + 1
),
maxphrase
AS
(
SELECT id, MAX(position) AS maxposition
FROM   phrases
GROUP  BY id
)
SELECT P.id, phrase + '.' AS PHRASE
FROM   phrases AS P
       INNER JOIN maxphrase AS M
             ON P.id = M.id
                AND P.position = M.maxposition
ORDER  BY id
```

Notez la façon récursive dont nous avons procédé pour reconstituer la phrase.

Chapitre 6

Exercices

Les exercices suivants permettent de mettre à jour plusieurs tables de la base exemple décrite au chapitre 1. On y trouvera l'ajout d'un nouvel enregistrement dans une table, la modification de colonnes et la suppression d'enregistrements.

EXERCICE 1 AJOUT

Énoncé

Ajoutez dans la table S_NEWS.T_UTILISATEUR_USR l'utilisateur « *Agnès Bidal* » ayant pour adresse e-mail *a.bidal@aol.com* et appartenant à l'organisation « *AOL* ». Affectez-lui le numéro suivant le plus grand identifiant de la table.

Solution

```
INSERT INTO S_NEWS.T_UTILISATEUR_USR
       (USR_ID, USR_MAIL,USR_TITRE,USR_NOM,USR_PRENOM,USR_ORGANISATION)
SELECT MAX(USR_ID) + 1,FROM S_NEWS.T_UTILISATEUR_USR
        'a.bidal@aol.com','Mlle','Bidal','Agnès','AOL'
```

EXERCICE 2 MODIFICATION

Énoncé

Modifiez l'utilisateur créé précédemment en mettant le nom en majuscules ; modifiez aussi son adresse e-mail en *a.bidal@aol.fr*.

Solution

```
UPDATE S_NEWS.T_UTILISATEUR_USR
SET    USR_NOM = UPPER(USR_NOM),
       USR_MAIL = 'a.bidal@aol.fr'
WHERE  USR_MAIL = 'a.bidal@aol.com',
       USR_TITRE = 'Mlle',
       USR_NOM = 'Bidal',
       USR_PRENOM = 'Agnès',
       USR_ORGANISATION = 'AOL'
```

EXERCICE 3 MODIFICATION

Énoncé

Modifiez le dernier utilisateur créé en mettant son nom en majuscules ; modifiez aussi son adresse e-mail en *a.bidal@club-Internet.fr*.

Solution

```
UPDATE S_NEWS.T_UTILISATEUR_USR
SET    USR_NOM = UPPER(USR_NOM),
       USR_MAIL = 'a.bidal@club-Internet.fr'
WHERE  USR_ID = (SELECT MAX(USR_ID) FROM S_NEWS.T_UTILISATEUR_USR)
```

Info

L'utilisation de la sous-requête SELECT MAX(USR_ID) FROM S_NEWS.T_UTILISATEUR_USR, dans le cas d'une clé auto-incrémentée, permet d'obtenir le dernier identifiant inséré, mais ne garantit en aucune façon que c'est celui que vous venez d'insérer. En effet, entre-temps, un autre utilisateur a pu insérer une ligne dans cette même table. Seule la connaissance de la valeur explicite de l'identifiant pour la ligne insérée garantit que l'on accède bien à la ligne considérée.

EXERCICE 4 AJOUT

Énoncé

Inscrivez l'utilisateur d'identifiant 3 auprès du fournisseur d'accès AOL.

Solution

```
INSERT INTO S_NEWS.T_J_ABONNE_ABN (USR_ID, FAI_ID)
VALUES (3, (SELECT FAI_ID FROM S_NEWS.T_FOURNINES_NEWS.T_FAI WHERE FAI_NOM = 'AOL'))
```

Une erreur aurait été d'écrire :

```
INSERT INTO S_NEWS.T_J_ABONNE_ABN (USR_ID, FAI_ID)
VALUES (3, 2)
```

En effet, rien ne permet de supposer que l'identifiant 2 restera toujours la valeur de la clé pour AOL.

EXERCICE 5 AJOUT

Énoncé

Ajoutez à la date/heure du jour l'inscription de l'utilisateur numéro 3 sur le forum numéro 5. Vérifiez en affichant le moment exact de l'inscription.

Solution

```
INSERT INTO S_NEWS.T_J_INSCRIT_ISC (USR_ID, FRM_ID, ISC_MOMENT)
       VALUES (3, 5, CURRENS_NEWS.T_TIMESTAMP)
SELECT USR_ID, FRM_ID, ISC_MOMENT
FROM   S_NEWS.T_J_INSCRIT_ISC
WHERE  USR_ID = 3 AND FRM_ID = 5
```

Pour Oracle, ces requêtes prennent la forme :

```
INSERT INTO S_NEWS.T_J_INSCRIT_ISC (USR_ID, FRM_ID, ISC_MOMENT)
       VALUES (3, 5, SYSDATE)
SELECT USR_ID, FRM_ID,TO_CHAR(ISC_MOMENT,'YYYY-MM-DD HH24:MI:SS')
FROM   S_NEWS.T_J_INSCRIT_ISC
WHERE  USR_ID = 3 AND FRM_ID = 5
```

SYSDATE est la fonction de récupération de la date du serveur, et TO_CHAR la conversion d'une date en littéral au format ISO.

Chapitre 6

EXERCICE 6 AJOUT

Énoncé

Ajoutez à la date du jour les deux news « *Prise en compte de XML ?* » et « *Persistance avec XML* », pour l'utilisateur numéro 3, postées sur le forum numéro 5 (la seconde news répondant à la première, une heure après). Pour cela, vous disposez de la fonction FN_GUID() qui permet d'obtenir un GUID (identificateur global unique). Ce GUID possède la valeur « *091A15A2-3CBA-4F24-B7E8-6816B8A2184E* ».

Solution

```
INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT MAX(NEW_ID) + 1, NULL, 3, 5,
       CURRENT_TIMESTAMP, F_GUID(), 'Prise en compte de XML ?'
FROM   S_NEWS.T_NEWS_NEW

INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT NEW_ID, 3, 5, CURRENT_TIMESTAMP + INTERVAL 1 HOUR,
       F_GUID(), 'Persistance avec XML.'
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_GLOBAL_ID = '091A15A2-3CBA-4F24-B7E8-6816B8A2184E'
```

Pour Oracle, ces requêtes prennent la forme :

```
INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT MAX(NEW_ID) + 1, NULL, 3, 5,
       SYSDATE, F_GUID(), 'Prise en compte de XML ?'
FROM   S_NEWS.T_NEWS_NEW
INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT NEW_ID, 3, 5, SYSDATE + 1/24, F_GUID(),'Persistance avec XML.'
FROM   S_NEWS.T_NEWS_NEW
WHERE  NEW_GLOBAL_ID = '091A15A2-3CBA-4F24-B7E8-6816B8A2184E'
```

Pour MS SQL Server, ces requêtes prennent la forme :

```
INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT MAX(NEW_ID) + 1, NULL, 3, 5,
       CURRENT_TIMESTAMP, F_GUID(), 'Prise en compte de XML ?')
FROM   S_NEWS.T_NEWS_NEW

INSERT INTO S_NEWS.T_NEWS_NEW (NEW_ID, NEW_ID_PERE, USR_ID, FRM_ID,
                               NEW_MOMENT, NEW_GLOBAL_ID, NEW_TITRE)
SELECT NEW_ID, 3, 5, DATEADD(HOUR, 1, CURRENT_TIMESTAMP),
       F_GUID(), 'Persistance avec XML.'
FROM   S_NEWS.T_NEWS_NEW
WHERE NEW_GLOBAL_ID = '091A15A2-3CBA-4F24-B7E8-6816B8A2184E'
```

EXERCICE 7 SUPPRESSION

Énoncé

Supprimez l'utilisateur numéro 3. Pour cet exercice, vous devez considérer les données de la base à l'origine. Si ce n'est pas le cas, supprimez la base de données et reconstruisez-la à l'aide des scripts SQL.

Solution

```
DELETE FROM S_NEWS.T_J_INSCRIT_ISC    WHERE USR_ID =3
DELETE FROM S_NEWS.T_J_ABONNE_ABN     WHERE USR_ID =3
DELETE FROM S_NEWS.T_NEWS_NEW         WHERE USR_ID =3
DELETE FROM S_NEWS.T_UTILISATEUR_USR  WHERE USR_ID =3
```

Vous devez impérativement faire attention à l'ordre des instructions DELETE de manière à préserver l'intégrité référentielle. En effet, toute tentative de suppression de ligne esclave, alors qu'un lien de référence persiste, provoquera le rejet de l'ordre et l'apparition d'un message d'erreur. Ainsi, la suppression de l'utilisateur avant celle des news qu'il a postées est impossible.

EXERCICE 8 SUPPRESSION CONDITIONNÉE

Énoncé

Quelle serait la condition à ajouter dans les instructions DELETE précédentes si on voulait supprimer les utilisateurs n'ayant jamais posté de news ? Quel est l'enchaînement des traitements ? Pour cet exercice, vous devez considérer les données de la base à l'origine. Si ce n'est pas le cas, supprimez la base de données et reconstruisez-la à l'aide des scripts SQL.

Solution

La condition à ajouter se base sur la liste suivante des identifiants d'utilisateurs qui n'ont pas posté de news :

```
SELECT USR_ID
FROM   S_NEWS.T_UTILISATEUR_USR USR
WHERE  NOT EXISTS (SELECT USR_ID
                   FROM S_NEWS.T_NEWS_NEW WHERE
                   USR_ID = USR.USR_ID)
```

Le script modifié devient donc le suivant :

```
DELETE FROM S_NEWS.T_J_INSCRIT_ISC
WHERE  USR_ID IN (SELECT USR_ID
                  FROM   S_NEWS.T_UTILISATEUR_USR USR
                  WHERE  NOT EXISTS (SELECT USR_ID
                                     FROM S_NEWS.T_NEWS_NEW WHERE
                                     USR_ID = USR.USR_ID))
DELETE FROM S_NEWS.T_J_ABONNE_ABN
WHERE  USR_ID IN (SELECT USR_ID
                  FROM   S_NEWS.T_UTILISATEUR_USR USR
                  WHERE  NOT EXISTS (SELECT USR_ID
                                     FROM S_NEWS.T_NEWS_NEW WHERE
                                     USR_ID = USR.USR_ID))
DELETE FROM S_NEWS.T_UTILISATEUR_USR
WHERE  USR_ID IN (SELECT USR_ID
```

```
                    FROM   S_NEWS.T_UTILISATEUR_USR USR
                    WHERE  NOT EXISTS (SELECT USR_ID
                                       FROM S_NEWS.T_NEWS_NEW WHERE
                                       USR_ID = USR.USR_ID))
```

Cette condition peut d'ailleurs s'exprimer par une jointure :

```
SELECT USR.USR_ID
FROM   S_NEWS.T_UTILISATEUR_USR USR
       LEFT OUTER JOIN S_NEWS.T_NEWS_NEW NEW
            ON USR.USR_ID = NEW.USR_ID
WHERE  NEW.USR_ID IS NULL
```

La première requête devient :

```
DELETE FROM S_NEWS.T_J_INSCRIT_ISC
WHERE  USR_ID IN (SELECT USR.USR_ID
                  FROM   S_NEWS.T_UTILISATEUR_USR USR
                         LEFT OUTER JOIN S_NEWS.T_NEWS_NEW NEW
                              ON USR.USR_ID = NEW.USR_ID
                  WHERE  NEW.USR_ID IS NULL)
```

D'autre part, on peut utiliser le prédicat EXISTS à la place du IN :

```
DELETE FROM S_NEWS.T_J_INSCRIT_ISC
WHERE  EXISTS (SELECT USR.USR_ID
               FROM   S_NEWS.T_UTILISATEUR_USR USR
                      LEFT OUTER JOIN S_NEWS.T_NEWS_NEW NEW
                           ON USR.USR_ID = NEW.USR_ID
               WHERE  NEW.USR_ID IS NULL
                 AND  USR.USR_ID = USR_ID)
```

La corrélation entre la requête cible du DELETE et la sous-requête est implicite du fait que la colonne USR_ID n'est préfixée par aucun alias.

EXERCICE 9 AJOUT CONDITIONNÉ

Énoncé

Vous ne savez pas si l'utilisateur d'identifiant 8 a été saisi. Dans le doute, écrivez la requête pour l'ajouter ou mettre à jour ses données avec les informations suivantes : identifiant 8, nom DUVAL, prénom Delphine, titre Mlle, adresse e-mail dd51@tiscali.fr.

Solution

Vous devez utiliser l'ordre MERGE pour y parvenir :

```
MERGE INTO S_NEWS.T_UTILISATEUR_USR AS U
USING (SELECT USR_ID
       FROM   S_NEWS.T_UTILISATEUR_USR
       WHERE  USR_ID = 8) AS UU
   ON U.USR_ID = UU.USR_ID)
WHEN MATCHED THEN
   UPDATE
   SET USR_NON    = 'DUVAL',
       USR_PRENOM = 'Delphine',
       USR_TITRE  = 'Mlle',
       USR_MAIL   = 'dd51@tiscali.fr'
WHEN NOT MATCHED THEN
   INSERT (USR_ID, USR_NON, USR_PRENOM, USR_TITRE, USR_MAIL)
   VALUES (8,      'DUVAL', 'Delphine', 'Mlle', 'dd51@tiscali.fr')
```

EXERCICE 10 AJOUT CONDITIONNÉ

Énoncé

Soit la table S_NEWS.T_TEST (COL1 INT, COL2 CHAR(4)), contenant les données suivantes :

```
COL1   COL2
------ ------
1      NORD
3      SUD
```

Écrivez une requête permettant d'insérer une ligne si cette ligne n'existe pas déjà dans la table.

Solution

Le principe est de faire un INSERT avec une sous-requête. La sous-requête compte le nombre de lignes qui comporte les mêmes données que la ligne à insérer. Si ce nombre est supérieur à zéro, alors la ligne existe déjà. Dans ce cas, il convient que la sous-requête ne retourne rien :

```
SELECT  1, 'NORD'
FROM    S_NEWS.T_TEST
WHERE   COL1 = 1
  AND   COL2 = 'NORD'
HAVING  COUNT(*) < 1
```

Ainsi le tuple n'est inséré que si les données n'existent pas. Pour certains SGBDR, on peut utiliser le NOT EXIST :

Solution pour Oracle :

```
SELECT  1, 'NORD'
FROM    DUAL
WHERE   NOT EXISTS (SELECT *
                    FROM    S_NEWS.T_TEST
                    WHERE   COL1 = 1
                      AND   COL2 = 'NORD')
```

Oracle utilise une pseudo-table de nom DUAL qui sert à remplir la clause FROM sans préciser un objet réellement existant (table ou vue).

Solution pour MS SQL Server :

```
SELECT  1, 'NORD'
WHERE   NOT EXISTS (SELECT *
                    FROM    S_NEWS.T_S_NEWS.T_S_NEWS.T_TEST
                    WHERE   COL1 = 1
                      AND   COL2 = 'NORD')
```

L'écriture pour SQL Server est encore plus sobre, car cet SGBDR ne requiert pas obligatoirement de clause FROM pour exécuter une telle requête.

Si votre SGBDR exige la clause FROM et ne propose pas de table "DUAL", alors créez-la de toutes pièces :

```
CREATE TABLE T (C INT);
INSERT INTO T VALUES (1);
```

Chapitre 7

Exercices

Les exercices d'application qui suivent consistent à écrire différents programmes sous la forme de fonctions ou procédures stockées. Quatre déclencheurs assurant diverses contraintes sont également à programmer.

EXERCICE 1 FONCTIONS

Énoncé

Écrivez la fonction F_FLIP, qui permute les éléments littéraux autour d'un caractère pivot, et F_REVERSE, qui renverse l'ordre des lettres :

```
F_FLIP('locomotive', 5) => motiveloco
F_FLIP('locomotive', 2) => ocomotivel
F_REVERSE('locomotive') => evitomocol
F_REVERSE('voiture')    => erutiov
```

Solution

```
CREATE FUNCTION F_FLIP ( mot VARCHAR(32), pivot INTEGER )
   RETURNS VARCHAR(32)
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_FLIP
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
RETURN CASE
          WHEN pivot <=1                     THEN mot
          WHEN pivot > CHARACTER_LENGTH(mot) THEN mot
          WHEN CHARACTER_LENGTH(mot) <= 1    THEN mot
          ELSE SUBSTRING(mot FROM pivot
                         FOR CHARACTER_LENGTH(mot) - pivot + 1)
             || SUBSTRING(mot FROM 1 FOR pivot - 1)
       END;

CREATE FUNCTION F_REVERSE ( mot VARCHAR(32) )
   RETURNS VARCHAR(32)
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_REVERSE
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
   IF CHARACTER_LENGTH(mot) <= 1
   THEN
      RETURN mot;
   END IF;
```

```
   DECLARE i INTEGER;
   DECLARE tom VARCHAR(32);
   SET i = 1;
   SET tom = '';
   REPEAT
      SET tom = tom || SUBSTRING(mot, i, 1);
      SET i = i +1;
   UNTIL i > CHARACTER_LENGTH(mot);
   RETURN tom;
END;
```

Solution pour MS SQL Server :

```
CREATE FUNCTION F_FLIP ( @mot VARCHAR(32), @pivot INTEGER )
   RETURNS VARCHAR(32)
BEGIN
   IF @mot IS NULL OR @pivot IS NULL
      RETURN NULL
   RETURN
      CASE
         WHEN @pivot <=1         THEN @mot
         WHEN @pivot > LEN(@mot) THEN @mot
          ELSE SUBSTRING(@mot, @pivot, LEN(@mot) - @pivot + 1) + SUBSTRING(@mot, 1,
@pivot - 1)
      END
   END
```

EXERCICE 2 FONCTIONS

Énoncé

Écrivez la fonction F_REPLACE, qui remplace un motif par un autre, et F_COUNS_NEWS.PATTERN, qui compte le nombre de motifs présents :

```
F_REPLACE('abracadabra', 'bra', 'zo') => azocadazo
F_REPLACE('borborygme', 'bor', 'tu')  => tutuygme
F_REPLACE('mimille', 'mil', '')       => mile
F_REPLACE('mimille', 'mi', '')        => lle
F_COUNS_NEWS.PATTERN('abracadabra', 'bra') => 2
F_COUNS_NEWS.PATTERN('Il faut qu''entre nous nous nous nourrissions', 'nou')
                                       => 4
```

Solution

```
CREATE FUNCTION F_REPLACE ( mot VARCHAR(32), old VARCHAR(32), new VARCHAR(32) )
   RETURNS VARCHAR(32)
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_REPLACE
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
-- gestion des effets de bord
   IF (CHARACTER_LENGTH(old) > CHARACTER_LENGTH(mot))
      OR (CHARACTER_LENGTH(new) > CHARACTER_LENGTH(mot))
   THEN
      RETURN mot;
   END IF
```

```
   DECLARE out VARCHAR(32);
   SET out = '';
   WHILE POSITION( old IN mot ) > 0
-- le motif à remplacer est situé au début du mot
      IF POSITION( old IN mot ) = 1
      THEN
         SET out = out + new;
         IF mot = old
         THEN
            SET mot = '';
         ELSE
            SET mot = SUBSTRING(mot FROM POSITION( old IN mot )
                                        + CHARACTER_LENGTH(old)
                                  FOR  CHARACTER_LENGTH(mot)
                                        - POSITION( old IN mot )
                                        - CHARACTER_LENGTH(old)
                                        + 1);
         END IF
      END IF
-- le motif à remplacer est situé à la fin du mot
      IF POSITION(old IN mot) + CHARACTER_LENGTH(old)
         = CHARACTER_LENGTH(mot)
      THEN
         SET out = out + SUBSTRING(mot FROM 1
                                       FOR POSITION( old IN mot ) -1)
                   + new;
         SET mot = '';
      END IF
-- le motif à remplacer est situé à l'intérieur du mot
      IF POSITION(old IN mot) BETWEEN 2
                              AND CHARACTER_LENGTH(mot)
                                  - CHARACTER_LENGTH(old)
                                  + 1
      THEN
         SET out = out + SUBSTRING(mot FROM 1
                                       FOR POSITION( old IN mot ) -1)
                   + new;
         SET mot = SUBSTRING(mot FROM POSITION( old IN mot )
                                     + CHARACTER_LENGTH(old)
                               FOR CHARACTER_LENGTH(mot)
                                     - POSITION( old IN mot )
                                     - CHARACTER_LENGTH(old) + 1);
      END IF
   END WHILE
   RETURN out + mot;
END;

CREATE FUNCTION F_COUNS_NEWS.PATTERN ( mot VARCHAR(32), motif VARCHAR(32))   RETURNS
INTEGER
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_COUNS_NEWS.PATTERN
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
-- gestion des effets de bord
   IF (CHARACTER_LENGTH(motif) > CHARACTER_LENGTH(mot))
      OR POSITION(motif IN mot) = 0
   THEN
      RETURN 0;
```

```
   END IF
-- comptage avec boucle
   DECLARE OUT INTEGER;
   SET OUT = 0;
   WHILE POSITION(motif IN mot) <> 0
      SET OUT = OUT + 1;
      IF CHARACTER_LENGTH(mot) > POSITION(motif IN mot)
                                 + CHARACTER_LENGTH(motif)
      THEN
         SET mot = SUBSTRING(mot, POSITION(motif IN mot)
                                  + CHARACTER_LENGTH(motif)
                                , CHARACTER_LENGTH(mot)
                                  - POSITION(motif IN mot)
                                  - CHARACTER_LENGTH(motif) + 1 );
      ELSE
         SET mot = '';
   END WHILE
   RETURN OUT;
END ;
```

Solution pour MS SQL Server :

```
CREATE FUNCTION F_COUNS_NEWS.PATTERN (@MOT VARCHAR(8000), @MOTIF VARCHAR(8000))
RETURNS INTEGER
AS
BEGIN
DECLARE @I INTEGER
-- cas trivial : donnée en entrée NULL
IF @MOT IS NULL OR @MOTIF IS NULL
BEGIN
   SET @I = NULL
   RETURN @I
END
-- cas trivial : donnée en entrée vide
IF @MOT ='' OR @MOTIF =''
BEGIN
   SET @I = 0
   RETURN @I
END
-- cas général
SET @I = 0
WHILE PATINDEX('%'+@MOTIF+'%', @MOT) > 0
BEGIN
   SET @I = @I +1
   IF LEN(@MOT) > PATINDEX('%'+@MOTIF+'%', @MOT) + LEN(@MOTIF)
      SET @MOT = SUBSTRING(@MOT, PATINDEX('%'+@MOTIF+'%', @MOT)
                                   + LEN(@MOTIF)
                               , LEN(@MOT)
                                 - PATINDEX('%'+@MOTIF+'%', @MOT)
                                 - LEN(@MOTIF)+1)
   ELSE
      SET @MOT = ''
END
RETURN @I
END
```

Chapitre 7

EXERCICE 3 FONCTIONS

Énoncé

Écrivez la fonction F_RESTRICT, qui restreint une chaîne aux seuls caractères passés en argument, et F_REPLACE_CHARS, qui remplace des caractères :

```
F_RESTRICT('à Paris ?', 'abcdefghijklmnopqrstuvwxyz')        => aris
F_RESTRICT('théâtre', 'abcdefghijklmnopqrstuvwxyz')          => thtre
F_RESTRICT('Airbus A320', '0123456789')                      => 320
F_REPLACE_CHARS('à Paris...?', 'abcdefghijklmnopqrstuvwxyz',
                               'ABCDEFGHIJKLMNOPQRSTUVWXYZ') => à PARIS ?
F_REPLACE_CHARS('théâtre', 'àâäéèêëîïôöùûüç',
                           'aaaeeeeiioouuuc')                => theatre
```

Solution

```
CREATE FUNCTION F_RESTRICT ( mot VARCHAR(32),
                             caracteres VARCHAR(32))
RETURNS INTEGER
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_RESTRICT
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
-- cas triviaux
   IF mot = '' OR caracteres = ''
   THEN
      RETURN mot;
   END IF
-- cas général
   DECLARE motout VARCHAR(32);
   SET motout = '';
   DECLARE i INTEGER;
   SET i = 1;
   WHILE i < CHARACTER_LENGTH(mot)
      IF POSITION(SUBSTRING(mot, i, 1) IN caracteres) = 0
      THEN
         SET motout = motout || SUBSTRING(mot, i, 1);
      ENDIF
      SET i = i +1;
   END WHILE
   RETURN motout;
END

CREATE FUNCTION F_REPLACE_CHARS ( mot      VARCHAR(32),
                                  cars_in  VARCHAR(32),
                                  cars_out VARCHAR(32) )
RETURNS INTEGER
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_REPLACE_CHARS
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
-- cas spécial
   IF CHARACTER_LENGTH(cars_in) <> CHARACTER_LENGTH(cars_out)
   THEN
      RETURN NULL;
-- cas triviaux
```

```
   IF mot = '' OR cars_in = ''
   THEN
      RETURN mot;
   END IF
-- cas général
   DECLARE motout VARCHAR(32);
   SET motout = '';
   DECLARE i INTEGER;
   SET i = 1;
   DECLARE car CHAR(1);
   DECLARE pos INTEGER;
   WHILE i < CHARACTER_LENGTH(mot)
      SET car = SUBSTRING(mot, i, 1);
      SET pos = POSITION(car IN cars_in);
      IF pos > 0
      THEN
         SET motout = motout || SUBSTRING(cars_out, pos, 1);
      ELSE
         SET motout = motout || car;
      ENDIF
      SET i = i +1;
   END WHILE
   RETURN motout;
END
```

Solution pour MS SQL Server :

```
CREATE FUNCTION F_REPLACE_CHARS (@mot VARCHAR (8000),
                                 @cars_in VARCHAR(256), @cars_out VARCHAR(256))
RETURNS VARCHAR (8000)
AS
BEGIN
-- effets de bord
   IF    @mot      IS NULL
      OR @cars_in  IS NULL
      OR @cars_out IS NULL
      OR LEN(cars_in) <> LEN(cars_out)
      RETURN NULL
   IF @mot) = '' OR cars_in = ''
      RETURN @mot
-- initialisation
   DECLARE @I INTEGER
   DECLARE @motout VARCHAR(8000)
   SET @motout = ''
-- lecture caractère par caractère
   SET @I =1
   WHILE @I <= LEN(@mot)
   BEGIN
      IF PATINDEX('%' + SUBSTRING(@mot, @I, 1)+ '%', @cars_in) > 0
      BEGIN
         IF LEN(@cars_out) >=
               PATINDEX('%' + SUBSTRING(@mot, @I, 1)+ '%', @cars_in)
            SET @motout = @motout + SUBSTRING(@cars_out, PATINDEX('%' +
                                    SUBSTRING(@mot, @I, 1)+ '%',
                                    @cars_in), 1)
      END
      ELSE
         SET @motout = @motout + SUBSTRING(@mot, @I, 1)
      SET @I = @I + 1
   END
   RETURN @motout
END
```

Chapitre 7

EXERCICE 4 FONCTIONS

Énoncé

Écrivez la fonction F_CONVERS_NEWS.HMS_HD, qui convertit une valeur de type TIME en heure décimale (FLOAT), et F_CONVERS_NEWS.HD_HMS, qui fait l'inverse. Vous pouvez, pour cela, créer d'autres fonctions.

Solution

```
CREATE FUNCTION F_CONVERS_NEWS.HMS_HD ( t TIME)
RETURNS FLOAT
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_CONVERS_NEWS.HMS_HD
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
RETURN CAST(EXTRACT(HOUR FROM t) AS FLOAT)
       + (CAST(EXTRACT(MINUTE FROM t) AS FLOAT) / 60.0)
       + (CAST(EXTRACT(SECOND FROM t) AS FLOAT) / 3600.0);

CREATE FUNCTION F_PADD_ZERO ( s VARCHAR(100), n SMALLINT)
RETURNS VARCHAR(100)
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_PADD_ZERO
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
   WHILE CHARACTER_LENGTH(s) < n
   DO
      SET s = '0' || s;
   END WHILE;
   RETURN s;
END

CREATE FUNCTION F_CONVERS_NEWS.HD_HMS ( t FLOAT)
RETURNS TIME
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC F_CONVERS_NEWS.HD_HMS
DETERMINISTIC
RETURN NULL ON NULL INPUT
CONTAINS SQL
BEGIN
   DECLARE H INTEGER;
   DECLARE M INTEGER;
   DECLARE S INTEGER;
   SET H = FLOOR(t);
   SET t = 60.0 * (t - FLOOR(t));
   SET M = FLOOR(t);
   SET t = 60.0 * (t - FLOOR(t));
   SET S = FLOOR(t);
   RETURN CAST(CAST(H AS VARCHAR(16)) || ':'
          || F_PADD_ZERO(CAST(M AS VARCHAR(2)), 2)
          || F_PADD_ZERO(CAST(S AS VARCHAR(2)), 2) AS TIME)
END
```

Pour cet exercice, il est préférable de réaliser une fonction de *padding* qui permet de compléter par des 0 placés en tête une chaîne de caractères correspondant à un entier,

afin de formater correctement le littéral en vue de sa conversion en type SQL TIME. C'est le sens de la fonction F_PADD_ZERO.

Solution pour MS SQL Server :

```
CREATE FUNCTION F_CONVERS_NEWS.HD_HMS (@HD FLOAT)
RETURNS VARCHAR(8)
AS
BEGIN
DECLARE @H INTEGER
DECLARE @M INTEGER
DECLARE @S INTEGER
DECLARE @RETVAL VARCHAR(8)
-- cas trivial
IF @HD IS NULL
   RETURN NULL
-- récupération des heures, minutes, secondes
SET @H  = FLOOR(@HD)
SET @HD = (@HD - @H) * 60
SET @M  = FLOOR(@HD)
SET @HD = (@HD - @M) * 60
SET @S  = FLOOR(@HD)
SET @RETVAL = CAST(@H AS CHAR(1))+':'
IF @M < 10
   SET @RETVAL = @RETVAL + '0' + CAST(@M AS CHAR(1))+':'
ELSE
   SET @RETVAL = @RETVAL + CAST(@M AS CHAR(2))+':'
IF @S < 10
   SET @RETVAL = @RETVAL + '0' + CAST(@S AS CHAR(1))
ELSE
   SET @RETVAL = @RETVAL + CAST(@S AS CHAR(2))+':'
RETURN @RETVAL
END
```

EXERCICE 5 FONCTION TABLE

Énoncé

Écrivez la fonction F_SEMAINE qui, à partir d'une date passée en argument, restitue les sept jours de la semaine correspondant à ladite date, sous forme de table.

Exemple :

```
SELECT *
FROM F_SEMAINE('2005-01-01')
JOUR_SEMAINE DATE_SEMAINE
------------ ------------
Lundi        2004-12-27
Mardi        2004-12-28
Mercredi     2004-12-29
Jeudi        2004-12-30
Vendredi     2004-12-31
Samedi       2005-01-01
Dimanche     2005-01-02
```

Solution

```
CREATE FUNCTION F_SEMAINE ( d DATE)
   RETURNS TABLE (JOUR_SEMAINE VARCHAR(8),
                  DATE_SEMAINE DATE)
   LANGUAGE SQL
   PARAMETER STYLE SQL
   SPECIFIC F_SEMAINE
```

```
   DETERMINISTIC
   MODIFIES SQL DATA
BEGIN
   WHILE MOD (CAST(d - DATE'2001-01-01' AS INTERVAL DAY), 6) + 1 <> 1
   DO
      SET d = d - INTERVAL 1 DAY;
   END WHILE;
   RETURN TABLE ( ('Lundi', d),
                  ('Mardi', d + INTERVAL 1 DAY),
                  ('Mercredi', d + INTERVAL 2 DAY),
                  ('Jeudi', d + INTERVAL 3 DAY),
                  ('Vendredi', d + INTERVAL 4 DAY),
                  ('Samedi', d + INTERVAL 5 DAY),
                  ('Dimanche', d + INTERVAL 6 DAY));
END;
```

SQL n'a pas implémenté de fonction de récupération d'un nom de jour à partir d'une date. Pour pallier le problème, il suffit de partir d'une date arbitraire dont vous connaissez le jour avec certitude : le 1er janvier 2001 est un lundi. Pour faire correspondre le jour de la semaine d'une date quelconque avec les valeurs 1 = lundi, 2 = mardi, 3 = mercredi… 7 = dimanche, vous pouvez effectuer le calcul suivant :

```
MOD (CAST (MA_DATE - DATE'2001-01-01' AS INTERVAL DAY, 6) + 1
```

L'algorithme est le suivant : dans un premier temps, remontez, en partant de la date passée en paramètre, jusqu'au premier lundi qui précède. Une fois ce lundi trouvé, recréez les dates depuis ce lundi jusqu'au dimanche en incrémentant la date courante de un jour.

EXERCICE 6 PROCÉDURE

Énoncé

Écrivez une procédure qui admet en paramètres deux entiers positifs et renvoie un jeu de résultats ne contenant que les nombres premiers compris entre les deux arguments. Le calcul sera réalisé sous forme procédurale puis sous forme ensembliste, dans deux procédures distinctes nommées P_PROC_PRIME et P_SETS_PRIME.

Solution

```
CREATE PROCEDURE P_PROC_PRIME
   ( IN minN INTEGER,
     IN maxN IN INTEGER )
   LANGUAGE SQL
   PARAMETER STYLE SQL
   SPECIFIC P_PROC_PRIME
   DETERMINISTIC
   MODIFIES SQL DATA
   DYNAMIC RESULT SETS 1
BEGIN
-- création d'une table temporaire pour stockage des nombres premiers
   CREATE LOCAL TEMPORARY TABLE TS_NEWS.PRIME_PRC (N INTEGER);
-- gestion des effets de bord : absence de valeur
   IF minN IS NULL OR maxN IS NULL
   THEN
      SELECT N FROM TS_NEWS.PRIME_PRC;
      RETURN;
```

```
   END IF;
-- gestion des effets de bord : valeurs négatives ou borne inversée
   IF minN < 0 OR minN > maxN
   THEN
      SELECT N FROM TS_NEWS.PRIME_PRC;
      RETURN;
   END IF;
-- déclaration des variables locales
   DECLARE diviseur INTEGER;
   DECLARE premier BOOLEAN;
-- boucle de minN à MaxN
   WHILE minN <= MaxN
   DO
-- on présuppose que la valeur courante est première
      SET premier = True;
      SET diviseur = 2;
-- boucle sur les diviseurs de la valeur courante n, de 2 à n-1
      WHILE diviseur < minN
      DO
-- s'il n'y a pas de reste, c'est que n n'est pas premier
         IF MOD(minN, diviseur) = 0
         THEN
            SET premier = False;
            LEAVE;
         END IF;
         SET diviseur = diviseur + 1;
      END WHILE;
-- si n est premier et différent de zéro, on l'insère dans la table
      IF premier AND minN <> 0
      THEN
         INSERT INTO TS_NEWS.PRIME_PRC VALUES (minN);
      END IF;
      SET minN = minN + 1;
   END WHILE;
-- renvoi le jeu de résultats comprenant les chiffres premiers
   SELECT N FROM TS_NEWS.PRIME_PRC;
END;

CREATE PROCEDURE P_SETS_PRIME
   ( IN minN INTEGER,
     IN maxN INTEGER )
   LANGUAGE SQL
   PARAMETER STYLE SQL
   SPECIFIC P_SETS_PRIME
   DETERMINISTIC
   MODIFIES SQL DATA
   DYNAMIC RESULT SETS 1
BEGIN
-- création d'une table temporaire pour stockage des nombres premiers
   CREATE LOCAL TEMPORARY TABLE TS_NEWS.PRIME_STS (N INTEGER);
-- gestion des effets de bord : absence de valeur
   IF minN IS NULL OR maxN IS NULL
   THEN
      SELECT N FROM TS_NEWS.PRIME_STS;
      RETURN;
   END IF;
-- gestion des effets de bord : valeurs négatives ou borne inversée
   IF minN < 0 OR minN > maxN
   THEN
      SELECT N FROM TS_NEWS.PRIME_STS;
      RETURN;
```

```
   END IF;
-- construction d'une table avec toutes les valeurs de 1 à maxN
   DECLARE i INTEGER;
   SET i = 1;
   WHILE i <= maxN
   BEGIN
      INSERT INTO TS_NEWS.PRIME_STS VALUES (i);
      SET i = @i + 1;
   END WHILE;
-- exécution de la requête extrayant les nombres premiers
   SELECT N
   FROM   TS_NEWS.PRIME_STS
   WHERE  N NOT IN (SELECT n1.N
                    FROM   TS_NEWS.PRIME_STS n1
                           JOIN TS_NEWS.PRIME_STS n2
                                 ON n2.N BETWEEN 2 AND n1.N - 1
                    WHERE  MOD(n1.N, n2.N) = 0
     AND N BETWEEN @minN AND @maxN;
END;
```

Comparez les vitesses d'exécution des deux procédures pour différentes valeurs de `minN` et `maxN`.

Solution pour MS SQL Server :

```
CREATE PROCEDURE P_PROC_PRIME
   @minN INTEGER,
   @maxN INTEGER
AS
BEGIN
   SET NOCOUNT ON
   CREATE TABLE #TS_NEWS.PRIME_PRC (N INTEGER)
   IF @minN IS NULL OR @maxN IS NULL
   BEGIN
      SELECT N FROM #TS_NEWS.PRIME_PRC
      RETURN
   END
   IF @minN < 0 OR @minN > @maxN
   BEGIN
      SELECT N FROM #TS_NEWS.PRIME_PRC
      RETURN
   END
   DECLARE @diviseur INTEGER
   DECLARE @premier BIT
   WHILE @minN < @maxN
   BEGIN
      SET @premier = 1
      SET @diviseur = 2
      WHILE @diviseur < @minN
      BEGIN
         IF @minN % @diviseur = 0
         BEGIN
            SET @premier = 0
            BREAK
         END
         SET @diviseur = @diviseur + 1
      END
      IF @premier = 1 AND @minN <> 0
      BEGIN
         INSERT INTO #TS_NEWS.PRIME_PRC VALUES (@minN)
      END
```

```
      SET @minN = @minN + 1
   END
   SELECT N FROM #TS_NEWS.PRIME_PRC
END

CREATE PROCEDURE P_SETS_PRIME
   @minN INTEGER,
   @maxN INTEGER
AS
BEGIN
   SET NOCOUNT ON
   CREATE TABLE #TS_NEWS.PRIME_STS (N INTEGER)
   IF @minN IS NULL OR @maxN IS NULL
   BEGIN
      SELECT N FROM TS_NEWS.PRIME_STS
      RETURN
   END
   IF @minN < 0 OR @minN > @maxN
   BEGIN
      SELECT N FROM #TS_NEWS.PRIME_STS
      RETURN
   END
   DECLARE @i INTEGER
   SET @i = 1
   WHILE @i <= @maxN
   BEGIN
      INSERT INTO #TS_NEWS.PRIME_STS VALUES (@i)
      SET @i = @i + 1
   END
   SELECT N
   FROM   #TS_NEWS.PRIME_STS
   WHERE  N NOT IN (SELECT n1.N
                    FROM   #TS_NEWS.PRIME_STS n1
                           JOIN #TS_NEWS.PRIME_STS n2
                                 ON n2.N BETWEEN 2 AND n1.N - 1
                    WHERE  n1.N % n2.N = 0)
     AND  N BETWEEN @minN AND @maxN
END
```

EXERCICE 7 PROCÉDURE

Énoncé

Écrivez une procédure qui insère une question dans un forum donné, avec ses éventuels fichiers, ou bien modifie les données. Le schéma possède la fonction F_GUID (), qui renvoie un GUID, et la fonction F_FILE_SIZE (`filePathName : varchar(128)`), qui renvoie la taille d'un fichier en octets. Les tables en jeu sont munies d'un auto-incrément de la clé.

Solution

```
CREATE PROCEDURE P_IU_QUESTION
 ( IN NEW_ID    INTEGER,
   IN USR_ID    INTEGER,
   IN FRM_ID    INTEGER,
   IN NEW_TITRE CHAR(256),
   IN NEW_TEXTE TEXT,
   IN FICS      ARRAY [] VARCHAR(128))
```

```
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC P_IU_QUESTION
DETERMINISTIC
MODIFIES SQL DATA
DYNAMIC RESULT SETS 0
BEGIN
-- si l'utilisateur ou le forum n'est pas mentionné, abandon de la procédure
   IF USR_ID IS NULL OR FRM_ID IS NULL
   THEN
      RETURN;
   END IF;
-- déclaration de la variable de récupération de l'état SQL
   DECLARE SQLSTATE CHAR(5);
-- pilotage de la transaction et de son niveau d'isolation
   SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
   START TRANSACTION;
-- si la clé de la news est indiquée, c'est une modification
   IF NEW_ID IS NOT NULL
   THEN
-- faire un UPDATE dans S_NEWS.NEWS_NEW :
      UPDATE S_NEWS.NEWS_NEW
      SET    USR_ID    = USR_ID,
             FRM_ID    = FRM_ID,
             NEW_TITRE = NEW_TITRE,
             NEW_TEXT  = NEW_TEXTE
      WHERE  NEW_ID = NEW_ID
      IF SQLSTATE >= '03000'
      THEN
         ROLLBACK TRANSACTION;
         RETURN;
      END IF;
-- supprimer les fichiers
      DELETE FROM S_NEWS.FICHIER_FIC
      WHERE  NEW_ID = NEW_ID
      IF SQLSTATE >= '03000'
      THEN
         ROLLBACK TRANSACTION;
         RETURN;
      END IF;
   ELSE
-- c'est une insertion, on calcule un GUID
      DECLARE GUID CHAR(36)
      SET GUID = F_GUID()
-- insérer dans la table S_NEWS.NEWS_NEW
      INSERT INTO S_NEWS.NEWS_NEW (USR_ID, FRM_ID, NEW_MOMENT,
                                   NEW_GLOBAL_ID, NEW_TITRE, NEW_TEXT)
                  VALUES     (USR_ID, FRM_ID, CURRENT_TIMESTAMP,
                                   GUID,          NEW_TITRE, NEW_TEXTE)
      IF SQLSTATE >= '03000'
      THEN
         ROLLBACK TRANSACTION;
         RETURN;
      END IF;
-- récupérer l'identifiant
      SET NEW_ID = (SELECT NEW_ID
                    FROM   S_NEWS.NEWS_NEW
                    WHERE  NEW_GLOBAL_ID = GUID)
      IF SQLSTATE >= '03000' OR NEW_ID IS NULL
      THEN
         ROLLBACK TRANSACTION;
```

```
         RETURN;
      END IF;
   END IF;
-- ajouter les nouveaux fichiers
   DECLARE i INTEGER;
   SET i = 1;
   WHILE i <= CARDINALITY(FICS)
   DO
      INSERT INTO S_NEWS.FICHIER_FIC (NEW_ID, FIC_NOM, FIC_TAILLE_O)
                        VALUES (NEW_ID, FICS[i], F_FILE_SIZE(FICS[i]))
      IF SQLSTATE >= '03000'
      THEN
         ROLLBACK TRANSACTION;
         RETURN;
      END IF;
      SET i = i + 1;
   END WHILE;
-- valider la transaction
   COMMIT TRANSACTION
END
```

Solution pour MS SQL Server :

```
CREATE PROCEDURE P_IU_QUESTION
   @NEW_ID    INTEGER,
   @USR_ID    INTEGER,
   @FRM_ID    INTEGER,
   @NEW_TITRE CHAR(256),
   @NEW_TEXTE TEXT,
   @FIC1      VARCHAR(128),
   @FIC2      VARCHAR(128),
   @FIC3      VARCHAR(128)
AS
BEGIN
-- si l'utilisateur ou le forum n'est pas mentionné, abandon de la procédure
   IF @USR_ID IS NULL OR @FRM_ID IS NULL
      RETURN
-- pilotage de la transaction et de son niveau d'isolation
   SET TRANSACTION ISOLATION LEVEL READ COMMITTED
   BEGIN TRANSACTION
-- si la clé de la news est indiquée, c'est une modification
   IF @NEW_ID IS NOT NULL
   BEGIN
-- faire un UPDATE dans S_NEWS.NEWS_NEW :
      UPDATE S_NEWS.NEWS_NEW
      SET    USR_ID    = @USR_ID,
             FRM_ID    = @FRM_ID,
             NEW_TITRE = @NEW_TITRE,
             NEW_TEXT  = @NEW_TEXTE
      WHERE  NEW_ID = @NEW_ID
      IF @@ERROR <> 0
         GOTO LBL_ERROR
-- supprimer les fichiers
      DELETE FROM S_NEWS.FICHIER_FIC
      WHERE  NEW_ID = @NEW_ID
      IF @@ERROR <> 0
         GOTO LBL_ERROR
   END
   ELSE
   BEGIN
-- c'est une insertion, on calcule un GUID
```

```
      DECLARE @GUID CHAR(36)
      SET @GUID = dbo.F_GUID()
-- insérer dans la table S_NEWS.NEWS_NEW
      INSERT INTO S_NEWS.NEWS_NEW (USR_ID,  FRM_ID,  NEW_MOMENT,
                                   NEW_GLOBAL_ID, NEW_TITRE,  NEW_TEXT)
                  VALUES     (@USR_ID, @FRM_ID, CURRENT_TIMESTAMP,
                                   @GUID,         @NEW_TITRE, @NEW_TEXTE)
      IF @@ERROR <> 0
         GOTO LBL_ERROR
-- récupérer l'identifiant
      SELECT @NEW_ID = NEW_ID
      FROM   S_NEWS.NEWS_NEW
      WHERE  NEW_GLOBAL_ID = @GUID
      IF @@ERROR <> 0
         GOTO LBL_ERROR
      IF @NEW_ID IS NULL
         GOTO LBL_ERROR
   END
-- ajouter les nouveaux fichiers
   IF @FIC1 IS NOT NULL
   BEGIN
      INSERT INTO S_NEWS.FICHIER_FIC (NEW_ID,  FIC_NOM,
                                      FIC_TAILLE_O)
                           VALUES (@NEW_ID, @FIC1,
                                      dbo.F_FILE_SIZE(@FIC1))
      IF @@ERROR <> 0
         GOTO LBL_ERROR
   END
   IF @FIC2 IS NOT NULL
   BEGIN
      INSERT INTO S_NEWS.FICHIER_FIC (NEW_ID,  FIC_NOM,
                                      FIC_TAILLE_O)
                           VALUES (@NEW_ID, @FIC2,
                                      dbo.F_FILE_SIZE(@FIC2))
      IF @@ERROR <> 0
         GOTO LBL_ERROR
   END
   IF @FIC3 IS NOT NULL
   BEGIN
      INSERT INTO S_NEWS.FICHIER_FIC (NEW_ID,  FIC_NOM,
                                      FIC_TAILLE_O)
                           VALUES (@NEW_ID, @FIC3,
                                      dbo.F_FILE_SIZE(@FIC3))
      IF @@ERROR <> 0
         GOTO LBL_ERROR
   END
-- valider la transaction
   COMMIT TRANSACTION
   RETURN
-- gestion du rollback en cas d'erreur
   LBL_ERROR:
   ROLLBACK TRANSACTION
END
```

Nous avons ici limité à 3 le nombre de fichiers passés dans la procédure pour SQL Server, ce dernier n'acceptant pas le type SQL ARRAY. Mais, dans un pareil cas, vous pouvez utiliser une table temporaire, qui remplace parfaitement un tableau.

EXERCICE 8 PROCÉDURE

Énoncé

Écrivez une procédure qui fournit la liste des colonnes littérales (CHAR, VARCHAR, NCHAR, NVARCHAR…) d'une table dont le nom est passé en argument. Une telle procédure a la forme :

```
P_LISTE_COLS_ALPHA ( IN nomSchema VARCHAR(128), IN nomTable VARCHAR(128), OUT listeCols VARCHAR (2048))
```

Utilisez la vue d'information de schéma INFORMATION_SCHEMA.COLUMNS et notamment des colonnes TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, DATA_TYPE (voir chapitre bonus, sur le site www.pearsoneducation.fr).

Exemple :

```
P_LISTE_COLS_ALPHA ( 'B_FORUM', 'S_NEWS.UUTILISATEUR_USR', listeCols)
```

Contenu de `listeCols` après exécution :

```
USR_MAIL, USR_TITRE, USR_NOM, USR_PRENOM, USR_ORGANISATION
```

Solution

```
CREATE PROCEDURE P_LISTE_COLS_ALPHA
 (  IN nomSchema VARCHAR(128),
    IN nomTable VARCHAR(128),
   OUT listeCols VARCHAR (2048))
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC P_LISTE_COLS_ALPHA
DETERMINISTIC
READS SQL DATA
DYNAMIC RESULT SETS 0
BEGIN
-- effets de bord : en cas d'absence de valeurs, on s'arrête là
   IF nomSchema IS NULL OR nomTable IS NULL
   THEN
      RETURN;
   END IF;
-- déclaration de la variable d'état d'exécution de SQL
DECLARE SQLSTATE CHAR(5);
-- variable de lecture de valeur de colonne
   DECLARE col VARCHAR(128);
-- création du curseur pour balayer les lignes du jeu de résultats
   DECLARE C_COLS INSENSITIVE NOT SCROLL CURSOR
   FOR
      SELECT COLUMN_NAME
      FROM   INFORMATION_SCHEMA.COLUMNS
      WHERE  TABLE_SCHEMA = @nomSchema
        AND  TABLE_NAME   = @nomTable
        AND  LOWER(DATA_TYPE) LIKE '%char%'
   FOR READ ONLY;
-- ouverture du curseur
   OPEN C_COLS;
-- lecture de la première ligne où est positionné le curseur
   FETCH C_COLS INTO col;
-- initialisation de la variable de sortie
   SET listeCols = '';
-- boucle de balayage du jeu de résultats avec le curseur
   WHILE SQLSTATE = '00000'
```

```
   DO
      SET listeCols = listeCols || ', ' || col;
      FETCH C_COLS INTO col;
   END
-- fermeture du curseur et désallocation mémoire
   CLOSE C_COLS;
-- suppression des caractères excédentaires
   IF listeCols IS NOT NULL
      SET listeCols = SUBSTRING(listeCols, 3, LEN(listeCols) - 2);
   END IF;
END
```

Solution pour Oracle :

```
CREATE OR REPLACE PROCEDURE p_liste_cols_alpha
   (nomSchema IN VARCHAR, nomTable IN VARCHAR, listeCols IN OUT VARCHAR)
IS
   CURSOR c IS SELECT COLUMN_NAME
                  FROM USER_TAB_COLUMNS
          WHERE TABLE_NAME = nomTable
          AND DATA_TYPE LIKE '%CHAR%';
BEGIN
   listeCols := '';
   FOR enreg IN c LOOP
          listeCols := listeCols || enreg.COLUMN_NAME || ', ';
   END LOOP;
   listeCols := SUBSTR(listeCols,1,LENGTH(listeCols)-2);
END  p_liste_cols_alpha;
```

Solution pour MS SQL Server :

```
CREATE PROCEDURE P_LISTE_COLS_ALPHA
   @nomSchema VARCHAR(128),
   @nomTable  VARCHAR(128),
   @listeCols VARCHAR (2048) OUTPUT
AS
BEGIN
-- effets de bord : en cas d'absence de valeurs, on s'arrête là
   IF @nomSchema IS NULL OR @nomTable IS NULL
      RETURN
-- variable de lecture de valeur de colonne
   DECLARE @col VARCHAR(128)
-- création du curseur pour balayer les lignes du jeu de résultats
   DECLARE C_COLS  CURSOR LOCAL STATIC READ_ONLY
   FOR
      SELECT COLUMN_NAME
      FROM   INFORMATION_SCHEMA.COLUMNS
      WHERE  TABLE_SCHEMA = @nomSchema
        AND  TABLE_NAME   = @nomTable
        AND  DATA_TYPE LIKE '%char%'
-- ouverture du curseur
   OPEN C_COLS
-- lecture de la première ligne où est positionné le curseur
   FETCH C_COLS INTO @col
-- initialisation de la variable de sortie
   SET @listeCols = ''
-- boucle de balayage du jeu de résultats avec le curseur
   WHILE @@FETCH_STATUS = 0
   BEGIN
      SET @listeCols = @listeCols + ', ' + @col
      FETCH C_COLS INTO @col
   END
```

```
-- fermeture du curseur et désallocation mémoire
   CLOSE C_COLS
   DEALLOCATE C_COLS
-- suppression des caractères excédentaires
   IF @listeCols IS NOT NULL
      SET @listeCols = SUBSTRING(@listeCols, 3, LEN(@listeCols) - 2)
END
```

Autre solution possible pour MS SQL Server, sans utilisation de curseur :

```
CREATE PROCEDURE P_LISTE_COLS_ALPHA
   @nomSchema VARCHAR(128),
   @nomTable  VARCHAR(128),
   @listeCols VARCHAR (2048) OUTPUT
AS
BEGIN
IF @nomSchema IS NULL OR @nomTable IS NULL
   RETURN
SET @listeCols = ''
SELECT @listeCols = @listeCols +', '+COLUMN_NAME
FROM   INFORMATION_SCHEMA.COLUMNS
WHERE  TABLE_SCHEMA = @nomSchema
  AND  TABLE_NAME   = @nomTable
  AND  DATA_TYPE LIKE '%char%'
IF @listeCols IS NOT NULL
   SET @listeCols = SUBSTRING(@listeCols, 3, LEN(@listeCols) - 2)
END
```

Cette solution se fonde sur le fait qu'une requête parcourt la table de manière linéaire et séquentielle. Dès lors, la concaténation des noms de colonnes se fait automatiquement dans la clause SELECT.

EXERCICE 9 PROCÉDURE

Énoncé

Écrivez une procédure capable de calculer le nombre d'e-mails par forum et par trimestre, et de fournir une synthèse pour une année passée en argument. Le jeu de résultats a la forme suivante :

```
FORUM         TRIM_1    TRIM_2    ... TRIM_4    CUMUL
------------- --------- --------- ... --------- ------------
Oracle        <null>    123       ... 456       9998
SQL           564       853       ... 742       56432
...           ...       ...       ... ...       ...
SQL Server    673       342       ... 897       11023
(TOUS)        98769     103765        19543     1500654
```

Pour cet exercice, créez une table temporaire à la volée (CREATE TEMPORAY LOCAL TABLE S_NEWS.CUMUL_FORUM_CMF AS… WITH DATA) et utilisez un curseur avec mise à jour. La ligne présentant le cumul de tous les forums doit venir en dernier.

Solution

```
CREATE PROCEDURE P_CUMUL_FORUM ( IN AN INTEGER )
LANGUAGE SQL
PARAMETER STYLE SQL
SPECIFIC P_CUMUL_FORUM
DETERMINISTIC
```

```
MODIFIES SQL DATA
DYNAMIC RESULT SETS 1
BEGIN
-- variable d'état SQL
DECLARE SQLSTATE CHAR(5);
-- création de la table et remplissage avec les données calculées
   CREATE TEMPORARY LOCAL TABLE S_NEWS.CUMUL_FORUM_CMF
   AS (SELECT FRM.FRM_ID, FRM_NOM AS FORUM,
              (SELECT COUNT(*)
               FROM   S_NEWS.NEWS_NEW NWD
               WHERE  NWD.FRM_ID = FRM.FRM_ID
                 AND  EXTRACT(MONTH FROM NEW_MOMENT) IN (1, 2, 3)
                 AND  EXTRACT(YEAR FROM NEW_MOMENT) = AN) AS TRIM_1,
              (SELECT COUNT(*)
               FROM   S_NEWS.NEWS_NEW NWD
               WHERE  NWD.FRM_ID = FRM.FRM_ID
                 AND  EXTRACT(MONTH FROM NEW_MOMENT) IN (4, 5, 6)
                 AND  EXTRACT(YEAR FROM NEW_MOMENT) = AN) AS TRIM_2,
              (SELECT COUNT(*)
               FROM   S_NEWS.NEWS_NEW NWD
               WHERE  NWD.FRM_ID = FRM.FRM_ID
                 AND  EXTRACT(MONTH FROM NEW_MOMENT) IN (7, 8, 9)
                 AND  EXTRACT(YEAR FROM NEW_MOMENT) = AN) AS TRIM_3,
              (SELECT COUNT(*)
               FROM   S_NEWS.NEWS_NEW NWD
               WHERE  NWD.FRM_ID = FRM.FRM_ID
                 AND  EXTRACT(MONTH FROM NEW_MOMENT) IN (10, 11, 12)
                 AND  EXTRACT(YEAR FROM NEW_MOMENT) = AN) AS TRIM_4,
              0 AS CUMUL
       FROM   S_NEWS.NEWS_NEW NEW
              INNER JOIN S_NEWS.FORUM_FRM FRM
                       ON NEW.FRM_ID = FRM.FRM_ID
       GROUP  BY FRM_NOM, FRM.FRM_ID
       UNION
-- dernière ligne pour cumul
       SELECT MAX(FRM_ID) + 1, '(TOUS)', 0, 0, 0, 0, 0
       FROM   S_NEWS.FORUM_FRM )
   WITH DATA;
-- variables locales pour balayage via curseur
   DECLARE FRM  CHAR(64),
           T1   INTEGER,
           T2   INTEGER,
           T3   INTEGER,
           T4   INTEGER,
           CL   INTEGER;
-- déclaration du curseur
   DECLARE C_CUMFRM CURSOR
   FOR
      SELECT FORUM, TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL
      FROM   S_NEWS.CUMUL_FORUM_CMF
      WHERE  FORUM <> '(TOUS)'
   FOR UPDATE OF TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL;
-- ouverture du curseur
   OPEN C_CUMFRM
-- récupération des valeurs de la première ligne
   FETCH C_CUMFRM INTO FRM, T1, T2, T3, T4, CL;
-- boucle de lecture des lignes du jeu de résultats avec le curseur
   WHILE SUBSTRING(SQLSTATE, 1, 2) = '00'
   DO
-- mise à jour de la table S_NEWS.CUMUL_FORUM_CMF
-- synchronisée avec la ligne sur laquelle est positionné le curseur
```

```
      UPDATE S_NEWS.CUMUL_FORUM_CMF
      SET    TRIM_1 = NULLIF(TRIM_1, 0),
             TRIM_2 = NULLIF(TRIM_2, 0),
             TRIM_3 = NULLIF(TRIM_3, 0),
             TRIM_4 = NULLIF(TRIM_4, 0),
             CUMUL  = T1 + T2 + T3 + T4
      WHERE  CURRENT OF C_CUMFRM;
      FETCH C_CUMFRM INTO FRM, T1, T2, T3, T4, CL;
   END WHILE;
-- fermeture du curseur
   CLOSE C_CUMFRM;
-- mise à jour de la dernière ligne (cumul pour tous les forums)
   UPDATE S_NEWS.CUMUL_FORUM_CMF
   SET    TRIM_1 = (SELECT SUM(TRIM_1) FROM S_NEWS.CUMUL_FORUM_CMF),
          TRIM_2 = (SELECT SUM(TRIM_2) FROM S_NEWS.CUMUL_FORUM_CMF),
          TRIM_3 = (SELECT SUM(TRIM_3) FROM S_NEWS.CUMUL_FORUM_CMF),
          TRIM_4 = (SELECT SUM(TRIM_4) FROM S_NEWS.CUMUL_FORUM_CMF)
   WHERE  FORUM = '(TOUS)';
-- mise à jour du cumul général
   UPDATE  S_NEWS.CUMUL_FORUM_CMF
   SET     CUMUL  = TRIM_1 + TRIM_2 + TRIM_3 + TRIM_4
   WHERE  FORUM = '(TOUS)';
-- restitution du résultat
   SELECT FORUM, TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL
   FROM   S_NEWS.CUMUL_FORUM_CMF
   ORDER BY FRM_ID;
END;
```

Pour la restitution du résultat, nous avons effectué un tri externe sur la colonne FRM_ID. Celle-ci a été alimentée avec les clés des différents forums. Pour la ligne cumul (TOUS), nous avons calculé la clé avec SELECT MAX(FRM_ID) + 1, afin d'obtenir la ligne des cumuls globaux en dernier dans le jeu de résultats.

Solution pour MS SQL Server :

```
CREATE PROCEDURE P_CUMUL_FORUM @AN INT
AS
SELECT FRM.FRM_ID, FRM_NOM AS FORUM,
       (SELECT COUNT(*)
        FROM   S_NEWS.NEWS_NEW NWD
        WHERE  NWD.FRM_ID = FRM.FRM_ID
          AND  MONTH(NEW_MOMENT) IN (1, 2, 3)
          AND  YEAR(NEW_MOMENT) = AN) AS TRIM_1,
       (SELECT COUNT(*)
        FROM   S_NEWS.NEWS_NEW NWD
        WHERE  NWD.FRM_ID = FRM.FRM_ID
          AND  MONTH(NEW_MOMENT) IN (4, 5, 6)
          AND  YEAR(NEW_MOMENT) = AN) AS TRIM_2,
       (SELECT COUNT(*)
        FROM   S_NEWS.NEWS_NEW NWD
        WHERE  NWD.FRM_ID = FRM.FRM_ID
          AND  MONTH(NEW_MOMENT) IN (7, 8, 9)
          AND  YEAR(NEW_MOMENT) = AN) AS TRIM_3,
       (SELECT COUNT(*)
        FROM   S_NEWS.NEWS_NEW NWD
        WHERE  NWD.FRM_ID = FRM.FRM_ID
          AND  MONTH(NEW_MOMENT) IN (10, 11, 12)
          AND  YEAR(NEW_MOMENT) = AN) AS TRIM_4,
       0 AS CUMUL
INTO   #S_NEWS.CUMUL_FORUM_CMF
FROM   S_NEWS.NEWS_NEW NEW
       INNER JOIN S_NEWS.FORUM_FRM FRM
```

```
            ON NEW.FRM_ID = FRM.FRM_ID
GROUP  BY FRM_NOM, FRM.FRM_ID
UNION
SELECT MAX(FRM_ID) + 1, '(TOUS)', 0, 0, 0, 0, 0
FROM   S_NEWS.FORUM_FRM
DECLARE @FORUM  CHAR(64),
        @T1     INTEGER,
        @T2     INTEGER,
        @T3     INTEGER,
        @T4     INTEGER,
        @CL     INTEGER
DECLARE C_CUMFRM CURSOR LOCAL
FOR
   SELECT FORUM, TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL
   FROM   #S_NEWS.CUMUL_FORUM_CMF
   WHERE  FORUM <> '(TOUS)'
FOR UPDATE OF TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL
OPEN C_CUMFRM
FETCH C_CUMFRM INTO @FORUM, @T1, @T2, @T3, @T4, @CL
WHILE @@FETCH_STATUS = 0
BEGIN
   UPDATE #S_NEWS.CUMUL_FORUM_CMF
   SET    TRIM_1 = NULLIF(TRIM_1, 0),
          TRIM_2 = NULLIF(TRIM_2, 0),
          TRIM_3 = NULLIF(TRIM_3, 0),
          TRIM_4 = NULLIF(TRIM_4, 0),
          CUMUL  = @T1 + @T2 + @T3 + @T4
   WHERE  CURRENT OF C_CUMFRM
   FETCH C_CUMFRM INTO @FORUM, @T1, @T2, @T3, @T4, @CL
END
CLOSE C_CUMFRM
DEALLOCATE C_CUMFRM
UPDATE  #S_NEWS.CUMUL_FORUM_CMF
SET       TRIM_1 = (SELECT SUM(TRIM_1) FROM #S_NEWS.CUMUL_FORUM_CMF),
          TRIM_2 = (SELECT SUM(TRIM_2) FROM #S_NEWS.CUMUL_FORUM_CMF),
          TRIM_3 = (SELECT SUM(TRIM_3) FROM #S_NEWS.CUMUL_FORUM_CMF),
          TRIM_4 = (SELECT SUM(TRIM_4) FROM #S_NEWS.CUMUL_FORUM_CMF)
   WHERE  FORUM = '(TOUS)'
UPDATE  #S_NEWS.CUMUL_FORUM_CMF
SET     CUMUL  = TRIM_1 + TRIM_2 + TRIM_3 + TRIM_4
   WHERE  FORUM = '(TOUS)'
SELECT FORUM, TRIM_1, TRIM_2, TRIM_3, TRIM_4, CUMUL
FROM #S_NEWS.CUMUL_FORUM_CMF
ORDER BY FRM_ID
```

EXERCICE 10 DÉCLENCHEUR

Énoncé

Écrivez, à l'aide d'un trigger, une contrainte d'unicité sur une colonne de type INTEGER qui ne tienne pas compte de la valeur 0 ou de l'absence de valeur. La table a pour nom S_NEWS.NUM et la colonne NUM_ID.

Solution

```
CREATE TRIGGER D_I_NUM_UNI
   BEFORE INSERT
   ON S_NEWS.NUM
BEGIN ATOMIC
   IF (SELECT MAX(C)
```

```
        FROM (SELECT COUNT(*) AS C
              FROM   S_NEWS.NUM
              WHERE  COALESCE(NUM_ID, 0) <> 0
              GROUP  BY NUM_ID)) > 1
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT = 'Violation de la contrainte d'unicité, '
                         || 'table S_NEWS.NUM, colonne NUM_ID';
   END IF;
END;
```

Mais ce trigger ne suffit pas à lui seul : il faut en écrire un autre ayant le même code, en cas de modification de la colonne NUM_ID.

```
CREATE TRIGGER D_U_NUM_UNI
   BEFORE UPDATE OF NUM_ID
   ON S_NEWS.NUM
…
```

Solution pour MS SQL Server :

```
CREATE TRIGGER D_IU_NUM_UNI
   ON S_NEWS.NUM
   FOR INSERT, UPDATE
AS
IF NOT UPDATE (UM_ID)
   RETURN
IF (SELECT MAX(C)
    FROM (SELECT COUNT(*) AS C
          FROM   S_NEWS.NUM
          WHERE  COALESCE(NUM_ID, 0) <> 0
          GROUP  BY NUM_ID) T) > 1
   ROLLBACK
```

MS SQL Server permet de combiner plusieurs triggers dans un même code. Il n'accepte pas les triggers BEFORE, mais implémente les triggers AFTER (par défaut) et INSTEAD. Pour annuler les effets d'un ordre SQL associé à un trigger AFTER, vous pouvez utiliser l'ordre ROLLBACK dans le corps du trigger.

EXERCICE 11 DÉCLENCHEUR : STANDARDISATION DE LA SAISIE DE DONNÉES

Énoncé

Soit la table de téléphone déclarée comme suit :

```
CREATE TABLE S_NEWS.TELEPHONE_TEL
(TEL_ID   INTEGER NOT NULL PRIMARY KEY,
 USR_ID   INTEGER NOT NULL
             FOREIGN KEY REFERENCES S_NEWS.UTILISATEUR_USR (USR_ID),
 TEL_NUM  VARCHAR(20) NOT NULL,
 TEL_MUN  CHAR(20))
```

Écrivez un trigger qui nettoie, lors de la mise à jour des données, les numéros de téléphone contenus dans la colonne TEL_NUM, en ne laissant que les chiffres et en les stockant à l'envers. Par exemple, le numéro de téléphone suivant 00~33 1 42-27-63-18 devient 8136722413300.

Solution

```
CREATE TRIGGER D_IU_TEL_NUM
   BEFORE INSERT
   ON S_NEWS.TELEPHONE_TEL REFERENCING NEW ROW AS NEW_TEL_ROW
   FOR EACH ROW
BEGIN ATOMIC
-- effet de bord : si la colonne est vide, ne rien faire !
```

```
   IF NEW_TEL_ROW.TEL_NUM IS NULL
   THEN
      RETURN;
   END IF;
-- initialisation du numéro inversé à vide
   SET NEW_TEL_ROW.TEL_NUM = '';
   DECLARE i INTEGER;
-- parcours à l'envers de la chaîne contenant le n° de téléphone
   SET i = CHARACTER_LENGTH(NEW_TEL_ROW.TEL_NUM);
   WHILE i >= 1
   DO
      IF SUBSTRING(NEW_TEL_ROW.TEL_NUM FROM i FOR 1)
            IN ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')
      THEN
-- ne retient que les chiffres
         SET SET NEW_TEL_ROW.TEL_NUM = NEW_TEL_ROW.TEL_NUM ||
                  SUBSTRING(NEW_TEL_ROW.TEL_NUM FROM i FOR 1);
      END IF;
      SET i = i-1;
   END WHILE;
END;
```

Solution pour MS SQL Server : SQL Server n'accepte pas le traitement par ligne (FOR EACH ROW) possible dans la norme SQL. Pour pallier cet inconvénient, vous pouvez par exemple créer une UDF qui réalise le traitement pour chaque ligne de la table. Utilisez ensuite cette UDF dans le cadre d'un ordre UPDATE du trigger.

```
-- fonction (UDF) de nettoyage de caractères non numériques
CREATE FUNCTION F_ONLYNUM (@data VARCHAR(256))
RETURNS VARCHAR(256)
AS
BEGIN
   IF @data IS NULL
      RETURN NULL
   IF RTRIM(@data) = ''
      RETURN ''
   DECLARE @out VARCHAR(256)
   DECLARE @i    INTEGER
   SET @data = RTRIM(LTRIM(@data))
   SET @i = 1
   SET @out = ''
   WHILE @i <= LEN(@data)
   BEGIN
      IF SUBSTRING(@data, @i, 1) BETWEEN '0' AND '9'
         SET @out = @out +  SUBSTRING(@data, @i, 1)
      SET @i = @i + 1
   END
   RETURN @out
END
-- le trigger
CREATE TRIGGER D_IU_TEL_NUM
   ON S_NEWS.TELEPHONE_TEL
   FOR INSERT, UPDATE
AS
IF NOT UPDATE (TEL_NUM)
   RETURN
UPDATE S_NEWS.TELEPHONE_TEL
SET    TEL_MUN = REVERSE(dbo.F_ONLYNUM(TEL_NUM))
WHERE  TEL_ID IN (SELECT TEL_ID FROM inserted)
```

Comme nous l'avons vu à l'exercice précédent, MS SQL Server n'accepte pas les triggers BEFORE, mais implémente les triggers AFTER (par défaut). Dans un tel cas, le trigger se déclenche de manière récursive puisqu'un ordre SQL UPDATE est lancé sur la table cible du trigger. Cependant, SQL Server dispose d'un paramétrage permettant de limiter le nombre d'appels récursifs des *nested triggers* (par défaut égal à 1).

EXERCICE 12 DÉCLENCHEUR : RÉALISATION D'UNE ASSERTION

Énoncé

Écrivez, à l'aide de déclencheurs, l'assertion de la section 3.5, qui oblige à l'unicité de la clé entre les tables S_NEWS.PROSPECS_NEWS.PSP et S_NEWS.CLIENS_NEWS.CLI :

```
CREATE ASSERTION A_UNIQUE_ID_CLI_PSP
CHECK (NOT EXISTS (SELECT *
                   FROM   S_NEWS.CLIENS_NEWS.CLI
                          JOIN S_NEWS.PROSPECS_NEWS.PSP
                               ON CLI_ID = PSP_ID))
```

Solution

Il faut vérifier qu'à l'insertion, l'identifiant d'un prospect n'est pas déjà attribué à un client, et *vice versa*, d'où deux triggers d'insertion :

```
CREATE TRIGGER D_I_PSP_KEY
   BEFORE INSERT
   ON S_NEWS.PROSPECS_NEWS.PSP REFERENCING NEW TABLE AS NEW_PSP_TBL
BEGIN ATOMIC
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.CLIENS_NEWS.CLI
       WHERE   CLI_ID IN (SELECT PSP_ID
                          FROM NEW_PSP_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité transversale.';
   END IF;
END;

CREATE TRIGGER D_I_CLI_KEY
   BEFORE INSERT
   ON S_NEWS.PROSPECS_NEWS.PSP REFERENCING NEW TABLE AS NEW_CLI_TBL
BEGIN ATOMIC
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.PROSPECS_NEWS.PSP
       WHERE   PSP_ID IN (SELECT CLI_ID
                          FROM NEW_CLI_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité transversale.';
   END IF;
END;
```

Mais cela ne suffit pas car il est toujours possible de changer la valeur d'un identifiant, d'où la nécessité de triggers complémentaires concernant les modifications :

```
CREATE TRIGGER D_U_PSP_KEY
   BEFORE UPDATE OF PSP_ID
   ON S_NEWS.PROSPECS_NEWS.PSP REFERENCING NEW TABLE AS NEW_PSP_TBL
```

```
BEGIN ATOMIC
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.CLIENS_NEWS.CLI
       WHERE   CLI_ID IN (SELECT PSP_ID
                          FROM NEW_PSP_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité transversale.';
   END IF;
END;
CREATE TRIGGER D_U_CLI_KEY
   BEFORE UPDATE OF CLI_ID
   ON S_NEWS.PROSPECS_NEWS.PSP REFERENCING NEW TABLE AS NEW_CLI_TBL
BEGIN ATOMIC
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.PROSPECS_NEWS.PSP
       WHERE   PSP_ID IN (SELECT CLI_ID
                          FROM NEW_CLI_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité transversale.';
   END IF;
END;
```

Ces triggers diffèrent uniquement par leur nature (INSERT/UPDATE). L'UPDATE se distingue également par le fait que seule la colonne clé est concernée.

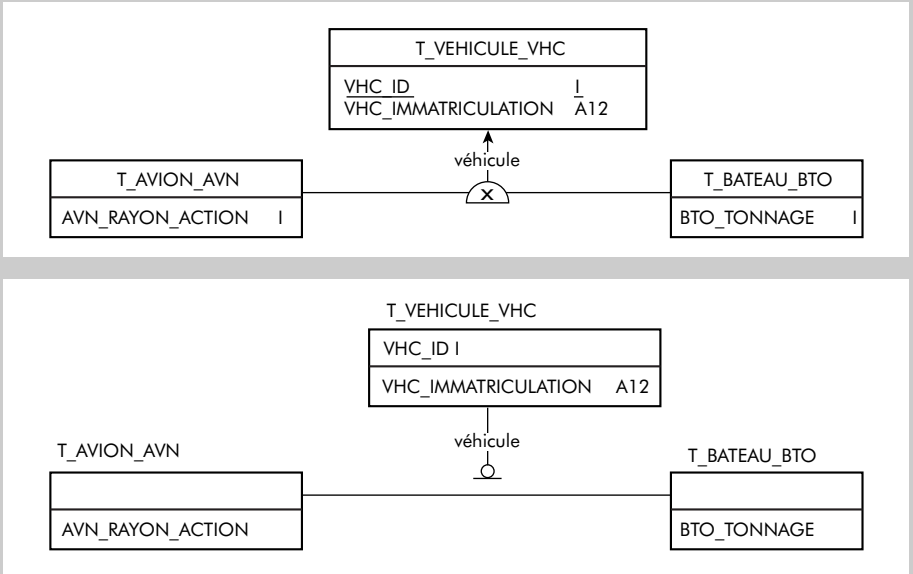
EXERCICE 13 DÉCLENCHEUR : EXCLUSIONS MUTUELLES ENTRE FILS D'UN HÉRITAGE

Énoncé

Écrivez les déclencheurs capables de gérer une exclusion mutuelle entre les tables du modèle en héritage des figures 7.2 et 7.3 :

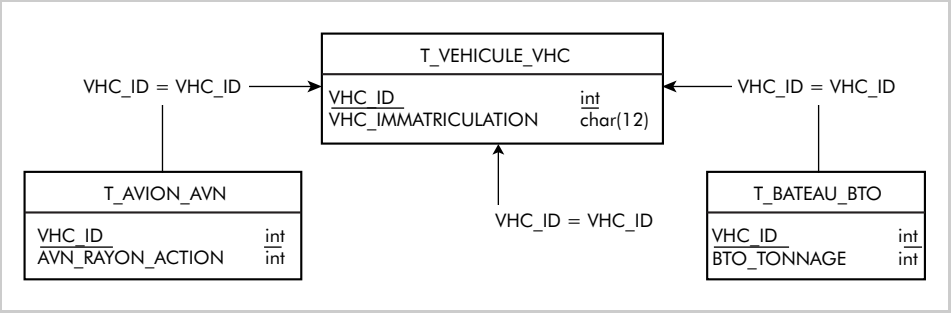
Figure 7.2 a et b

Modèles conceptuels MERISE et UML.



Énoncé *(suite)*

Figure 7.3
Modèles logiques.



Solution

Le principe est le suivant : avant toute création dans une branche, vérifiez que l'identifiant n'est pas déjà utilisé par un fils. En cas de modification de l'identifiant, vérifiez qu'il n'a pas déjà été utilisé par un fils. Lors de la suppression d'un fils, supprimez-le aussi dans la table mère. Dans le cas des deux fils du modèle ci-dessus, cela conduit à écrire six déclencheurs :

```
CREATE TRIGGER D_I_AVN_FILS
   BEFORE INSERT
   ON S_NEWS.AVION_AVN REFERENCING NEW TABLE AS NEW_AVN_TBL
BEGIN ATOMIC
-- on vérifie si l'identifiant n'est pas déjà attribué au bateau
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.BATEAU_BTO
       WHERE   VHC_ID IN (SELECT VHC_ID
                          FROM   NEW_AVN_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité d'héritage de véhicule.';
   END IF;
END;

CREATE TRIGGER D_I_BTO_FILS
   BEFORE INSERT
   ON S_NEWS.BATEAU_BTO REFERENCING NEW TABLE AS NEW_BTO_TBL
BEGIN ATOMIC
-- on vérifie si l'identifiant n'est pas déjà attribué à un avion
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.AVION_AVN
       WHERE   VHC_ID IN (SELECT VHC_ID
                          FROM   NEW_BTO_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité d'héritage de véhicule.';
   END IF;
END;

CREATE TRIGGER D_U_AVN_FILS
   BEFORE UPDATE OF AVN_ID
   ON S_NEWS.AVION_AVN REFERENCING NEW TABLE AS NEW_AVN_TBL
BEGIN ATOMIC
```

```
-- on vérifie si l'identifiant n'est pas déjà attribué au bateau
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.BATEAU_BTO
       WHERE   VHC_ID IN (SELECT VHC_ID
                          FROM   NEW_AVN_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité d'héritage de véhicule.';
   END IF;
END;

CREATE TRIGGER D_U_BTO_FILS
   BEFORE INSERT OF BTO_ID
   ON S_NEWS.BATEAU_BTO REFERENCING NEW TABLE AS NEW_BTO_TBL
BEGIN ATOMIC
-- on vérifie si l'identifiant n'est pas déjà attribué à un avion
   IF (SELECT  COUNT(*)
       FROM    S_NEWS.AVION_AVN
       WHERE   VHC_ID IN (SELECT VHC_ID
                          FROM   NEW_BTO_TBL) > 0)
   THEN
      RESIGNAL SQLSTATE '70000';
      SET MESSAGE_TEXT =
          'Violation de contrainte d'unicité d'héritage de véhicule.';
   END IF;
END;

CREATE TRIGGER D_D_AVN_FILS
   BEFORE DELETE
   ON S_NEWS.AVION_AVN REFERENCING OLD TABLE AS OLD_AVN_TBL
BEGIN ATOMIC
-- on supprime dans la table mère
   DELETE FROM S_NEWS.VEHICULE_VHC
   WHERE  VHC_ID IN (SELECT VHC_ID
                     FROM   OLD_AVN_TBL)
END;

CREATE TRIGGER D_D_BTO_FILS
   BEFORE DELETE
   ON S_NEWS.BATEAU_BTO REFERENCING OLD TABLE AS OLD_BTO_TBL
BEGIN ATOMIC
-- on supprime dans la table mère
   DELETE FROM S_NEWS.VEHICULE_VHC
   WHERE  VHC_ID IN (SELECT VHC_ID
                     FROM   OLD_BTO_TBL)
END;
```

EXERCICE 14 DÉCLENCHEUR : GESTION D'UNE ANTI-RELATION

Énoncé

Écrivez les triggers nécessaires à la gestion d'une « anti-relation »… Soient la table des mots indexés :

```
CREATE TABLE S_NEWS.MOS_NEWS.INDEX_MDX
(MDX_ID   INTEGER NOT NULL PRIMARY KEY,
 MDX_MOT  CHAR(32) NOT NULL)
```

Énoncé *(suite)*

et la table des « mots noirs » (dépourvu en quelque sorte de contenu sémantique, comme « le », « la », « les », « un », « une », « des »…) :

```
CREATE TABLE S_NEWS.MOS_NEWS.NOIR_MNR
(MNR_MOT  CHAR(32) NOT NULL PRIMARY KEY)
```

Les triggers doivent interdire l'indexation d'un mot noir dans la table S_NEWS.MOS_NEWS.INDEX_MDX et le placement d'un mot « non noir » dans la table S_NEWS.MOS_NEWS.NOIR_MNR.

Solution

```
CREATE TRIGGER D_I_MDX
   AFTER INSERT
   ON S_NEWS.MOS_NEWS.INDEX_MDX REFERENCING NEW ROW AS NEW_MDX_ROW
   FOR EACH ROW
BEGIN ATOMIC
-- si le mot existe déjà en tant que mot noir, on le suprimme
   IF (SELECT COUNT(*)
       FROM   S_NEWS.MOS_NEWS.NOIR_MNR
       WHERE  MNR_MOT = NEW_MDX_ROW.MDX_MOT)
   THEN
      DELETE FROM S_NEWS.MOS_NEWS.INDEX_MDX
      WHERE  MDX_MOT = NEW_MDX_ROW.MDX_MOT;
   END IF;
END;

CREATE TRIGGER D_I_MNR
   AFTER INSERT
   ON S_NEWS.MOS_NEWS.NOIR_MNR REFERENCING NEW ROW AS NEW_MNR_ROW
   FOR EACH ROW
BEGIN ATOMIC
-- si le mot noir existe déjà en tant que mot indexé, on le suprimme
   IF (SELECT COUNT(*)
       FROM   S_NEWS.MOS_NEWS.INDEX_MDX
       WHERE  MDX_MOT = NEW_MNR_ROW.MNR_MOT)
   THEN
      DELETE FROM S_NEWS.MOS_NEWS.NOIR_MNR
      WHERE  MNR_MOT = NEW_MNR_ROW.MNR_MOT;
   END IF;
END;
```

Chapitre 8

Exercices

Les exercices suivants décrivent la gestion d'utilisateurs au sein de la base de données exemple décrite au chapitre 1.

EXERCICE 1 UTILISATEURS

Énoncé

Créez un utilisateur ayant pour identifiant `userFAI` et un autre ayant pour identifiant `userForum`.

Solution

Solution pour Oracle :

```
--Création des utilisateurs
CREATE USER    userFAI
 IDENTIFIED BY userFAI
 DEFAULT    TABLESPACE USERS
 TEMPORARY  TABLESPACE TEMP
 QUOTA      UNLIMITED  ON USERS;
CREATE USER    userForum
 IDENTIFIED BY userForum
 DEFAULT    TABLESPACE USERS
 TEMPORARY  TABLESPACE TEMP
 QUOTA      UNLIMITED  ON USERS;
-- Attribution des autorisations d'accès
GRANT CONNECT, RESOURCE TO userFAI, userForum;
```

Solution pour MS SQL Server :

```
-- création des utilisateurs avec connexion au serveur
EXEC master.dbo.sp_addlogin 'userFAI', 'passwordUserFAI', 'B_FORUM'
GO
EXEC master.dbo.sp_addlogin 'userForum', 'passwordUserForum', 'B_FORUM'
GO
-- positionnement sur la base B_FORUM
USE B_FORUM
GO
-- autorisation d'accès des utilisateurs à la base B_FORUM
EXEC sp_grantdbaccess 'userFAIm'
GO
EXEC sp_grantdbaccess 'userForum'
GO
```

EXERCICE 2 RÔLES

Énoncé

Créez les trois rôles suivants : `gestFAI`, `gestUtil` et `gestForum`.

Solution

```
CREATE ROLE gestFAI
CREATE ROLE gestUtil
CREATE ROLE gestForum
```

EXERCICE 3 ALIMENTATION DES RÔLES

Énoncé

Alimentez les trois rôles créés dans le cadre de l'exercice précédent, en vous fondant sur le tableau présenté ci-dessous, qui décrit les privilèges (S pour SELECT, I pour INSERT, U pour UPDATE et D pour DELETE) à définir pour les tables indiquées :

Table	gestFAI	gestUtil	gestForum
S_NEWS.UTILISATEUR_USR	S	SIUD	S
S_NEWS.SERVEUR_SRV	SIUD		
S_NEWS.NEWS_NEW		S	SIUD
S_NEWS.FOURNINES_NEWS.FAI	SIUD		
S_NEWS.FORUM_FRM		S	SIUD
S_NEWS.FICHIER_FIC			SIUD
TJ_RECENSE_RCS	SIUD	S	
TJ_INSCRIS_NEWS.ISC		SIUD	S
TJ_ABONNE_ABN	SIUD	S	

Solution

```
GRANT SELECT                          ON S_NEWS.UTILISATEUR_USR TO gestFAI
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.SERVEUR_SRV     TO gestFAI
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.FOURNINES_NEWS.FAI   TO gestFAI
GRANT SELECT, INSERT, UPDATE, DELETE ON TJ_RECENSE_RCS    TO gestFAI
GRANT SELECT, INSERT, UPDATE, DELETE ON TJ_ABONNE_ABN     TO gestFAI
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.UTILISATEUR_USR TO gestUtil
GRANT SELECT                          ON S_NEWS.NEWS_NEW        TO gestUtil
GRANT SELECT                          ON S_NEWS.FORUM_FRM       TO gestUtil
GRANT SELECT                          ON TJ_ABONNE_ABN     TO gestUtil
GRANT SELECT, INSERT, UPDATE, DELETE ON TJ_INSCRIS_NEWS.ISC    TO gestUtil
GRANT SELECT                          ON S_NEWS.UTILISATEUR_USR TO gestForum
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.NEWS_NEW        TO gestForum
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.FORUM_FRM       TO gestForum
GRANT SELECT, INSERT, UPDATE, DELETE ON S_NEWS.FICHIER_FIC     TO gestForum
GRANT SELECT                          ON TJ_RECENSE_RCS    TO gestForum
GRANT SELECT                          ON TJ_INSCRIS_NEWS.ISC    TO gestForum
```

EXERCICE 4 RÔLES/UTILISATEURS

Énoncé

Affectez les rôles `gestFAI`, `gestUtil` à l'utilisateur `userFAI`, et les rôles `gestUtil`, `gestForum` à l'utilisateur `userForum`.

Solution

```
--rôles gestFAI, gestUtil à userFAI
GRANT gestFAI, gestUtil TO userFAI;
--rôles gestUtil, gestForum à userForum
GRANT gestUtil, gestForum TO userForum;
```

Chapitre 8

EXERCICE 5 PRÉROGATIVES

Énoncé

Soient les utilisateurs usrAdmFrm et usrAdm qui sont des administrateurs de bases de données devant « nettoyer » les données du forum (lecture, insertion, suppression). Donnez-leur les privilèges nécessaires au nettoyage des données de la table des news, mais aussi pour créer des tables temporaires identiques à celle des news de façon à pouvoir copier les données tout en maintenant l'intégrité référentielle.

Solution

```
GRANT SELECT, INSERT, DELETE
      ON S_NEWS.NEWS_NEW
      TO usrAdmFrm, usrAdm
GRANT REFERENCES
      ON S_NEWS.UTILISATEUR_USR
      TO usrAdmFrm, usrAdm
GRANT REFERENCES
      ON S_NEWS.FORUM_FRM
      TO usrAdmFrm, usrAdm
```

Pour pouvoir créer une table, même temporaire, en utilisant des références d'intégrité à d'autres tables *via* les clés étrangères (lien d'intégrité référentielle), les utilisateurs doivent être en mesure de faire référence aux tables concernées et donc disposer des privilèges de référence sur les tables S_NEWS.UTILISATEUR_USR et S_NEWS.FORUM_FRM.

EXERCICE 6 INFORMATIONS DE SCHÉMAS

Énoncé

Écrivez une requête qui, à partir de la vue d'information de schéma INFORMATION_SCHEMA.TABLE_PRIVILEGES, et en particulier des colonnes GRANTEE, TABLE_NAME, PRIVILEGE_TYPE et IS_GRANTABLE, permet de reconstituer les ordres de création des privilèges (voir chapitre bonus sur le site www.pearsoneducation.fr). Le contenu de cette vue est le suivant :

```
GRANTEE        TABLE_NAME           PRIVILEGE_TYPE IS_GRANTABLE
-------------- -------------------- -------------- ------------
usrAdmFrm      S_NEWS.FORUM_FRM          REFERENCES     NO
usrAdmFrm      S_NEWS.NEWS_NEW           SELECT         YES
usrAdmFrm      S_NEWS.NEWS_NEW           INSERT         NO
usrAdmFrm      S_NEWS.NEWS_NEW           DELETE         NO
usrAdm         S_NEWS.NEWS_NEW           SELECT         NO
usrAdm         S_NEWS.UTILISATEUR_USR    REFERENCES     NO
```

Solution

```
SELECT 'GRANT '  || PRIVILEGE_TYPE
       || ' ON ' || TABLE_NAME
       || ' TO ' || GRANTEE
       || CASE IS_GRANTABLE
            WHEN 'NO' THEN ''
            ELSE ' WITH GRANT OPTION'
          END AS DCL_ORDER
FROM   INFORMATION_SCHEMA.TABLE_PRIVILEGES
```

Notez que cette requête n'est pas suffisante pour redéfinir tous les privilèges. En effet, certains d'entre eux peuvent porter sur des colonnes de table ou de vue. Pour les redéfinir, il faudrait aussi utiliser la vue d'information de schéma INFORMATION _SCHEMA.COLUMN_PRIVILEGES. Dans ce cas, l'utilisation d'une procédure stockée serait plus adaptée.