Interrogation de bases de données SQL

stph.scenari-community.org/bdd
sql2.pdf

Stéphane Crozat

19 janvier 2018

Table des matières

Cours

SQL* est un langage standardisé, implémenté par tous les SGBDR*, qui permet, indépendamment de la plate-forme technologique et de façon déclarative, de définir le modèle de données, de le contrôler et enfin de le manipuler.

1. Questions simples avec le Langage de Manipulation de Données (SELECT)

1.1. Exercice : Représentation de représentants

[30 minutes]

Soit le schéma relationnel et le code SQL suivants :

```
REPRESENTANTS (#NR, NOMR, VILLE)
PRODUITS (#NP, NOMP, COUL, PDS)
CLIENTS (#NC, NOMC, VILLE)
VENTES (#NR=>REPRESENTANTS(NR), #NP=>PRODUITS(NP), #NC=>CLIENTS(NC), QT)
```

```
/* Les requêtes peuvent être testées dans un SGBDR, en créant une base de
données avec le script SQL suivant */

/*
DROP TABLE VENTES ;
DROP TABLE CLIENTS ;
DROP TABLE PRODUITS ;
DROP TABLE REPRESENTANTS ;
*/

CREATE TABLE REPRESENTANTS (
  NR INTEGER PRIMARY KEY,
  NOMR VARCHAR,
  VILLE VARCHAR
);

CREATE TABLE PRODUITS (
  NP INTEGER PRIMARY KEY,
  NOMP VARCHAR,
  COUL VARCHAR,
  PDS INTEGER
);

CREATE TABLE CLIENTS (
  NC INTEGER PRIMARY KEY,
  NOMC VARCHAR,
  VILLE VARCHAR
);

CREATE TABLE VENTES (
  NR INTEGER REFERENCES REPRESENTANTS(NR),
  NP INTEGER REFERENCES PRODUITS(NP),
  NC INTEGER REFERENCES CLIENTS(NC),
  QT INTEGER,
  PRIMARY KEY (NR, NP, NC)
);

INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (1, 'Stephane', 'Lyon');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (2, 'Benjamin', 'Paris');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (3, 'Leonard', 'Lyon');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (4, 'Antoine', 'Brest');
INSERT INTO REPRESENTANTS (NR, NOMR, VILLE) VALUES (5, 'Bruno', 'Bayonne');

INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (1, 'Aspirateur', 'Rouge', 3546
);
```

```
44 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (2, 'Trottinette', 'Bleu', 1423
   );
45 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (3, 'Chaise', 'Blanc', 3827);
46 INSERT INTO PRODUITS (NP, NOMP, COUL, PDS) VALUES (4, 'Tapis', 'Rouge', 1423);
47
48 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (1, 'Alice', 'Lyon');
49 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (2, 'Bruno', 'Lyon');
50 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (3, 'Charles', 'Compiègne');
51 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (4, 'Denis', 'Montpellier');
52 INSERT INTO CLIENTS (NC, NOMC, VILLE) VALUES (5, 'Emile', 'Strasbourg');
53
54 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 1, 1);
55 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 1, 2, 1);
56 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (2, 2, 3, 1);
57 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (4, 3, 3, 200);
58 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 2, 190);
59 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (1, 3, 2, 300);
60 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 2, 120);
61 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 4, 120);
62 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 4, 2);
63 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 1, 3);
64 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 4, 1, 5);
65 INSERT INTO VENTES (NR, NP, NC, QT) VALUES (3, 1, 3, 1);
```

Écrire en SQL les requêtes permettant d'obtenir les informations ci-après.

Question 1

[solution n°1 p.28]

Tous les détails de tous les clients.

Question 2

[solution n°2 p.28]

Les numéros et les noms des produits de couleur rouge et de poids supérieur à 2000.

Question 3

[solution n°3 p.28]

Les représentants ayant vendu au moins un produit.

Question 4

[solution n°4 p.28]

Les noms des clients de Lyon ayant acheté un produit pour une quantité supérieure à 180.

Question 5

[solution n°5 p.29]

Les noms des représentants et des clients à qui ces représentants ont vendu un produit de couleur rouge pour une quantité supérieure à 100.

1.2. Question (SELECT)

Fondamental : Question

La requête de *sélection* ou *question* est la base de la recherche de données en SQL.

Définition : Sélection

La sélection est la composition d'un *produit cartésien*, d'une *restriction* et d'une *projection* (ou encore la composition d'une jointure et d'une projection).

Syntaxe

```
SELECT liste d'attributs projetés
FROM liste de relations du produit cartésien
WHERE condition de la restriction
```

- La partie SELECT indique le sous-ensemble des attributs qui doivent apparaître dans la réponse (c'est le schéma de la relation résultat).
- La partie FROM décrit les relations qui sont utilisables dans la requête (c'est à dire l'ensemble des attributs que l'on peut utiliser).
- La partie WHERE exprime les conditions que doivent respecter les attributs d'un tuple pour pouvoir être dans la réponse. Une condition est un prédicat et par conséquent renvoie un booléen. Cette partie est optionnelle.

Exemple

```
SELECT Nom, Prenom
FROM Personne
WHERE Age>18
```

Cette requête sélectionne les attributs Nom et Prénom des tuples de la relation Personne, ayant un attribut Age supérieur à 18.

Syntaxe : Notation préfixée

Afin de décrire un attribut d'une relation en particulier (dans le cas d'une requête portant sur plusieurs relations notamment), on utilise la notation `relation.attribut`.

Exemple

```
SELECT Personne.Nom, Personne.Prenom, Vol.Depart
FROM Personne, Vol
WHERE Personne.Vol=Vol.Numero
```

*Syntaxe : SELECT **

Pour projeter l'ensemble des attributs d'une relation, on peut utiliser le caractère * à la place de la liste des attributs à projeter.

Exemple

```
SELECT *
FROM Avion
```

Cette requête sélectionne tous les attributs de la relation Avion.

Notons que dans cet exemple, la relation résultat est exactement la relation Avion

1.3. Opérateurs de comparaisons et opérateurs logiques

Introduction

La clause WHERE d'une instruction de sélection est définie par une condition. Une telle condition s'exprime à l'aide d'opérateurs de comparaison et d'opérateurs logiques. Le résultat d'une expression de condition est toujours un booléen.

Définition : Condition

```
1 Condition Elémentaire ::= Propriété <Opérateur de comparaison> Constante
2 Condition ::= Condition <Opérateur logique> Condition | Condition Elémentaire
```

Les opérateurs de comparaison sont :

- P = C
- P <> C
- P < C
- P > C
- P <= C
- P >= C
- P BETWEEN C1 AND C2
- P IN (C1, C2, ...)
- P LIKE 'chaîne'
- P IS NULL

Les opérateur logique sont :

- OR
- AND
- NOT

Remarque : Opérateur LIKE

L'opérateur `LIKE` 'chaîne' permet d'insérer des jokers dans l'opération de comparaison (alors que l'opérateur `=` teste une égalité stricte) :

- Le joker % désigne 0 ou plusieurs caractères quelconques
- Le joker _ désigne 1 et 1 seul caractère

On préférera l'opérateur `=` à l'opérateur `LIKE` lorsque la comparaison n'utilise pas de joker.

1.4. Renommage de colonnes et de tables avec les alias

Syntaxe : Alias de table

Il est possible de redéfinir le nom des relations au sein de la requête afin d'en simplifier la syntaxe.

```
1 SELECT t1.attribut1, t2.attribut2
2 FROM table1 t1, table2 t2
```

Exemple

```
1 SELECT Parent.Prenom, Enfant.Prenom
2 FROM Parent, Enfant
3 WHERE Enfant.Nom=Parent.Nom
```

Cette requête sélectionne les prénoms des enfants et des parents ayant le même nom. On remarque la notation Parent.Nom et Enfant.Nom pour distinguer les attributs Prenom des relations Parent et Enfant.

On notera que cette sélection effectue une jointure sur les propriétés Nom des relations Parent et Enfant.

Remarque : Alias d'attribut (AS)

Il est possible de redéfinir le nom des propriétés de la relation résultat.

```
1 SELECT attribut1 AS a1, attribut2 AS a2
2 FROM table
```

1.5. Dédoublonnage (SELECT DISTINCT)

Attention : SELECT DISTINCT

L'opérateur SELECT n'élimine pas les doublons (i.e. les tuples identiques dans la relation résultat) par défaut. Il faut pour cela utiliser l'opérateur SELECT DISTINCT.

Exemple

```
1 SELECT DISTINCT Avion
2 FROM Vol
3 WHERE Date=31-12-2000
```

Cette requête sélectionne l'attribut Avion de la relation Vol, concernant uniquement les vols du 31 décembre 2000 et renvoie les tuples sans doublons.

1.6. Tri (ORDER BY)

Introduction

On veut souvent que le résultat d'une requête soit trié en fonction des valeurs des propriétés des tuples de ce résultat.

Syntaxe : ORDER BY

```
1 SELECT liste d'attributs projetés
2 FROM liste de relations
3 WHERE condition
4 ORDER BY liste ordonnée d'attributs
```

Les tuples sont triés d'abord par le premier attribut spécifié dans la clause ORDER BY, puis en cas de doublons par le second, etc.

Remarque : Tri décroissant

Pour effectuer un tri décroissant on fait suivre l'attribut du mot clé "DESC".

Exemple

```
1 SELECT *
2 FROM Personne
3 ORDER BY Nom, Age DESC
```

1.7. Projection de constante

Syntaxe : Projection de constante

Il est possible de projeter directement des constantes (on utilisera généralement un alias d'attribut pour nommer la colonne).

```
1 SELECT constante AS nom
```

Cette requête renverra une table avec une seule ligne et une seule colonne à la valeur de *constante*.

Exemple

```
1 SELECT '1' AS num
```

```
1  num
2 -----
3  1
```

Exemple

```
1 SELECT 'Hello world' AS hw
```

```
1      hw
2 -------------
3  Hello world
```

Exemple

```
1 SELECT CURRENT_DATE AS now
```

```
1     now
2 ------------
3  2016-10-21
```

1.8. Commentaires en SQL

L'introduction de commentaires au sein d'une requête SQL se fait :

- En précédant les commentaires de `--` pour les commentaires mono-ligne
- En encadrant les commentaires entre `/*` et `*/` pour les commentaires multi-lignes.

Exemple

```
/*
* Creation of table Student
* Purpose : managing information about students in the system
*/
CREATE TABLE person (
pknum VARCHAR(13) PRIMARY KEY, --Student national number
name VARCHAR(25)
);


```

2. Opérations d'algèbre relationnelle en SQL

2.1. Expression d'une restriction

Syntaxe

```
SELECT *
FROM R
WHERE <condition>
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R

A	B	C
1	Alpha	10
2	Bravo	10

```
Restriction(R1,C<20)

SELECT *
FROM R1
WHERE C<20
```

Exemple de restriction (SQL et Algèbre)

2.2. Expression d'une projection

Syntaxe

```
SELECT P1, P2, Pi
FROM R
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R

A	C
1	10
2	10
3	20
4	

```
Projection(R1,A,C)

SELECT A, C
FROM R1
```

Exemple de projection (SQL et Algèbre)

2.3. Expression du produit cartésien

Syntaxe

```
1 SELECT *
2 FROM R1, R2, Ri
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

A	B	C	X	Y
1	Alpha	10	10	Echo
1	Alpha	10	20	Fox
1	Alpha	10	30	Golf
2	Bravo	10	10	Echo
2	Bravo	10	20	Fox
2	Bravo	10	30	Golf
3	Charlie	20	10	Echo
3	Charlie	20	20	Fox
3	Charlie	20	30	Golf
4	Delta		10	Echo
4	Delta		20	Fox
4	Delta		30	Golf

```
Produit(R1,R2)

SELECT *
FROM R1,R2
```

Exemple de produit (SQL et Algèbre)

2.4. Expression d'une jointure

Syntaxe : Jointure par la clause WHERE

En tant que composition d'un produit cartésien et d'une restriction la jointure s'écrit :

```
1 SELECT *
2 FROM R1, R2, Ri
3 WHERE <condition>
```

Avec `Condition` permettant de joindre des attributs des `Ri`

Syntaxe : Jointure par la clause ON

On peut également utiliser la syntaxe dédiée suivante :

```
1 SELECT *
2 FROM R1 INNER JOIN R2
3 ON <condition>
```

Et pour plusieurs relations :

```
1 SELECT *
2 FROM (R1 INNER JOIN R2 ON <condition>) INNER JOIN Ri ON <condition>
3
```

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox

```
Jointure(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 INNER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure (SQL et Algèbre)

Exemple : Une jointure naturelle

```
1 SELECT *
2 FROM R1, R2
3 WHERE R2.NUM = R1.NUM
```

Remarque : Auto-jointure

Pour réaliser une auto-jointure, c'est à dire la jointure d'une relation avec elle-même, on doit utiliser le renommage des relations. Pour renommer une relation, on note dans la clause FROM le nom de renommage après le nom de la relation : "FROM NOM_ORIGINAL NOUVEAU_NOM".

Exemple : Auto-jointure

```
SELECT E1.Nom
FROM Employe E1, Employe E2
WHERE E1.Nom= E2.Nom
```

2.5. Exercice : Tableau final

[solution n°6 p.29*]*

Que renvoie la dernière instruction SQL de la séquence ci-dessous ?

```
1 CREATE TABLE R (a INTEGER, b INTEGER);
2 INSERT INTO R VALUES (1,1);
3 INSERT INTO R VALUES (1,2);
4 INSERT INTO R VALUES (3,3);
5 SELECT * FROM R R1, R R2 WHERE R1.a=R2.a;
```

☐

1	1
1	2
3	3

☐

1	1
1	1
1	2
1	2
3	3
3	3

☐

1	1	1	1
1	1	1	2
1	1	3	3
1	2	1	1
1	2	1	2
1	2	3	3
3	3	1	1
3	3	1	2
3	3	3	3

☐

1	1	1	1
1	1	1	2
1	2	1	1
1	2	1	2
3	3	3	3

☐

1	1	1	2
1	1	3	3
1	2	1	1
1	2	3	3
3	3	1	1
3	3	1	2

2.6. Expression d'une jointure externe

Syntaxe : Jointure externe, gauche ou droite

Pour exprimer une jointure externe on se base sur la syntaxe INNER JOIN en utilisant à la place OUTER JOIN, LEFT OUTER JOIN ou RIGHT OUTER JOIN.

Exemple : Jointure externe gauche

```
1 SELECT Num
2 FROM Avion LEFT OUTER JOIN Vol
3 ON Avion.Num=Vol.Num
```

Cette requête permet de sélectionner tous les avions, y compris ceux non affectés à un vol.

Remarque

Remarquons que "Avion LEFT OUTER JOIN Vol" est équivalent à "Vol RIGHT OUTER JOIN Avion" en terme de résultat.

Intuitivement, on préfère utiliser la jointure gauche pour sélectionner tous les tuple du côté N d'une relation 1: N, même si il ne sont pas référencés ; et la jointure droite pour pour sélectionner tous les tuples d'une relation 0: N, y compris ceux qui ne font pas de référence. Cette approche revient à toujours garder à gauche de l'expression "JOIN" la relation "principale", i.e. celle dont on veut tous les tuples, même s'ils ne référencent pas (ou ne sont pas référencés par) la relation "secondaire".

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	**Delta**	

R2

#X	Y
10	Echo
20	Fox
30	**Golf**

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
4	**Delta**			
			30	**Golf**

```
JointureExterne(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 OUTER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure externe (SQL et Algèbre)

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	**Delta**	

R2

#X	Y
10	Echo
20	Fox
30	Golf

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
4	**Delta**			

```
JointureExterneGauche(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 LEFT OUTER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure externe gauche (SQL et Algèbre)

Exemple

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	**Golf**

R

A	B	C	X	Y
1	Alpha	10	10	Echo
2	Bravo	10	10	Echo
3	Charlie	20	20	Fox
			30	**Golf**

```
JointureExterneDroite(R1,R2,R1.C=R2.X)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
```

Exemple de jointure externe droite (SQL et Algèbre)

Méthode : Trouver les enregistrements non joints

La jointure externe sert en particulier pour trouver les enregistrements d'une table qui ne sont pas référencés par une clé étrangère. Il suffit de sélectionner, après la jointure externe, tous les enregistrements pour lesquels la clé de la relation référençante est nulle, on obtient alors ceux de la relation référencée qui ne sont pas référencés.

R1

#A	B	C=>R2
1	Alpha	10
2	Bravo	10
3	Charlie	20
4	Delta	

R2

#X	Y
10	Echo
20	Fox
30	**Golf**

R

A	B	C	X	Y
			30	**Golf**

```
Restriction(
        JointureExterneDroite(R1,R2,R1.C=R2.X),
        R1.A IS NULL)

SELECT *
FROM R1 RIGHT OUTER JOIN R2
ON R1.C=R2.X
WHERE R1.A IS NULL
```

Exemple de sélection d'enregistrements non référencés (SQL et Algèbre)

2.7. Exercice : Photos à gauche

[solution n°7 p.30]

Soit les trois relations instanciées suivantes :

Numero	Nom	Prenom
1	Jacques	Brel
2	Georges	Brassens
3	Léo	Ferré

Relation Personne

Personne	Photo
1	1
2	1
3	1

Relation PersonnesPhotos

Numero	Date	Lieu
1	10/12/1965	Paris
2	18/03/2002	Lille

Relation Photo

Combien de tuples renvoie l'instruction SQL suivante :

```
SELECT *
FROM Photo ph LEFT JOIN (
 PersonnesPhotos pp INNER JOIN Personne pe
  ON pp.Personne=pe.Numero
) ON ph.Numero=pp.Photo;
```

2.8. Opérateurs ensemblistes

Introduction

Les opérateurs ensemblistes ne peuvent être exprimés à l'aide de l'instruction de sélection seule.

Syntaxe : Union

```
SELECT * FROM R1
UNION
SELECT * FROM R2
```

Syntaxe : Intersection

```
SELECT * FROM R1
INTERSECT
SELECT * FROM R2
```

Syntaxe : Différence

```
1 SELECT * FROM R1
2 EXCEPT
3 SELECT * FROM R2
```

Remarque

Les opérations INTERSECT et EXCEPT n'existe que dans la norme SQL2, et non dans la norme SQL1. Certains SGBD sont susceptibles de ne pas les implémenter.

Exercices

II

1. Exercice : Location d'appartements

[20 min]

Soit le schéma relationnel suivant gérant le fonctionnement d'une agence de location d'appartements.

```
APPARTEMENT(#code_appt:String, adresse:String, type:{studio,F1,F2,F3,F4,F5+},
prix_loyer:Real)
LOCATAIRE(#code_loc:String, nom:String, prenom:String)
LOCATION(#code_loc=>Locataire, #code_appt=>Appartement)
PAIEMENT_LOYER(#code_loc=>Locataire, #code_appt=>Appartement, #date_payement:
Date, prix_paye:Real)
```

Question 1

[solution n°8 p.31*]*

En algèbre relationnelle et en SQL afficher tous les paiements effectués par un locataire avec le code X.

Question 2

[solution n°9 p.31*]*

En algèbre relationnelle et en SQL afficher l'adresse de l'appartement loué par un locataire avec le code X.

Question 3

[solution n°10 p.32*]*

En algèbre relationnelle et en SQL proposer une requête qui affiche tous les appartements libres.

2. Exercice : Employés et salaires

[20 minutes]

Soit le schéma relationnel :

```
Employe (#Num, Nom, Prenom, Age, Salaire, Fonction=>Fonction, Societe=>Societe)
Fonction (#Intitule, SalaireMin, SalaireMax, NbHeures)
Societe (#Nom, Pays, Activite)
```

Question 1

[solution n°11 p.32*]*

Écrivez une requête SQL permettant de sélectionner les noms de tous les directeurs de France.

Question 2

[solution n°12 p.32*]*

Écrivez une requête SQL permettant d'afficher le salaire de tous les employés en francs (sachant que le salaire dans la table est en euros et que un euro vaut 6.55957 franc).

Question 3

[solution n°13 p.32*]*

Écrivez une requête SQL permettant de vérifier que les salaires de chaque employé correspondent bien à ce qui est autorisé par leur fonction.

Question 4

[solution n°14 p.32*]*

Écrivez une requête SQL permettant le passage aux 35 heures :

- en modifiant le nombre d'heures travaillées pour chaque fonction (ramené à 35 s'il est supérieur à 35),
- et en réajustant les échelles de salaire au pro-rata du nombre d'heures travaillées avant le passage aux 35 heures.

Devoirs

1. Exercice : Library

[30 min]

Soit le modèle relationnel suivant :

```
1 author (aid, aname)
2 book (bid, title, category)
3 student (sid, sname, dept)
4 write (aid, bid)
5 borrow (sid, bid)
```

- *author* représente une table d'auteurs. Chaque ligne contient le nom et l'identifiant d'un auteur
- *book* représente une table de livres. Chaque ligne est un livre décrit par son identifiant, son titre et sa catégorie (roman, science-fiction, musique, etc.).
- *student* représente une table d'étudiants. Chaque ligne est un étudiant décrit par son identifiant, son nom et son département (informatique, mécanique...).
- *write* représente l'association entre les auteurs et les livres. Une ligne de cette table signifie que l'auteur *aid* a écrit le livre *bid*
- *borrow* représente les informations de prêt de livre. Une ligne de cette table signifie que l'étudiant *sid* a emprunté le livre *bid*, à la date *checkout-time* et l'a retourné à la date *return-time*.
- Tous les attributs sont des chaînes de caractères, sauf *checkout-time* et *return-time* qui sont des *timestamps* (la concaténation d'une date et d'un temps dans la journée en heures, minutes et secondes).
- Pour *author*, *book*, *student*, tous les identifiants sont des clés. L'attribut *title* d'un livre est une clé. Les relations *write* et *borrow* expriment chacune une association N:M.
- Deux auteurs ou deux étudiants peuvent avoir le même nom.
- Un étudiant ne peut pas emprunter deux fois le même livre.

Question 1

Réécrire le modèle relationnel correctement, en intégrant les clés (primaires, candidates et étrangères), et les types de données.

Question 2

Rétro-concevoir le modèle MCD en UML

Indice :

Classe d'association *(cf. p.25)*

Question 3

Écrire en algèbre relationnel les requêtes suivantes :

- Trouver les titres de tous les livres que l'étudiant sid='S15' a emprunté
- Trouver les titres de tous les livres qui n'ont jamais été empruntés par un étudiant

Question 4

Écrire en SQL les requêtes suivantes :

- Trouver tous les étudiants qui ont emprunté le livre bid='B30'
- Trouver les titres de tous les livres empruntés par des étudiants en informatique (*dept* = 'informatique')

2. Exercice : Gestion de bus

[30 min]

Une société de gestion d'autobus urbains dispose de la base de données suivante pour représenter ses conducteurs, ses bus et ses lignes de bus :

```
1 Conducteur (#matricule:varchar(10), nom:varchar, prénom:varchar)
2 Ligne (#num:numeric(2), km:numeric(4))
3 Bus (#immat:varchar(10), type:{Soufflets|Normaux|Réduits}, kilométrage:numeric
  (7), matricule=>Conducteur)
4 TrajetParJour (#ligne=>Ligne, #immat=>Bus, nombre_de_trajets:numeric(2))
5 Station (#num:numeric, nom:varchar, adresse:varchar)
6 Arrêt (#ligne:=>Ligne, #station=>Station, rang:numeric)
```

- Un conducteur est identifié par son matricule.
- Une ligne de bus est représentée par un numéro et elle a une longueur exprimée en kilomètres.
- Un bus est conduit par un seul conducteur référencé par son matricule. Le type du bus peut comprendre les valeurs Soufflets, Normaux, Réduits, le kilométrage correspond au nombre de kilomètres au compteur du bus.
- Trajet permet de connaître le nombre de trajets effectués par un bus sur une ligne donnée pendant une journée.
- Station contient la liste des stations desservies par la société.
- Arrêt fait le lien entre les lignes et les stations qu'elle dessert. Le rang correspond au numéro de l'arrêt su la ligne (le rang 1 correspond au départ de la ligne, l'arrivée est la station qui a le rang le plus élevé sur cette ligne).

Donnez en algèbre relationnelle et en SQL les requêtes permettant d'obtenir les informations suivantes.

Question 1

Nom et prénom du conducteur du bus immatriculé "5657 MA 49".

Question 2

Immatriculation et type des bus qui partent de la station "Gare SNCF".

Question 3

Numéros des lignes desservant à la fois les stations "Royallieu" et "Gare SNCF".

Question 4

Numéros des lignes qui ne sont pas desservies par des bus à soufflets.

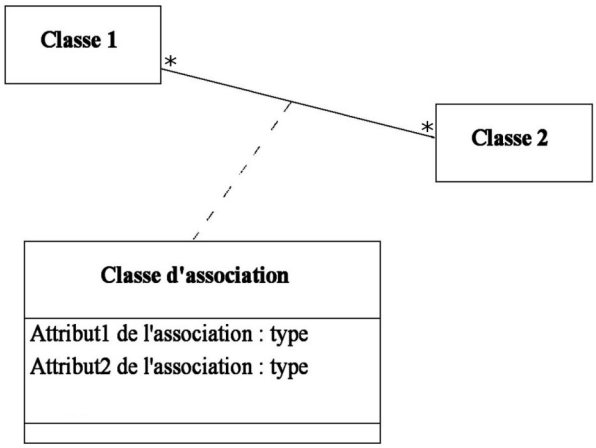
Contenus annexes

> Classe d'association

Définition : Classe d'association

On utilise la notation des classes d'association lorsque l'on souhaite ajouter des propriétés à une association.

Syntaxe : Notation d'une classe d'association en UML



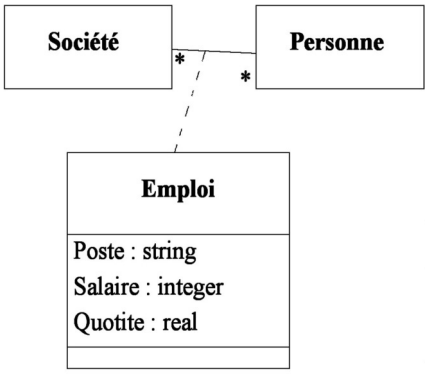
Notation d'une classe d'association en UML

Méthode

On réserve en général les classes d'association aux associations N:M.

Il est toujours possible de réduire une classe d'association sur une association 1:N en migrant ses attributs sur la classe côté N, et c'est en général plus lisible ainsi.

Exemple : Exemple de classe d'association



Emplois

Conseil

Selon le standard UML une classe d'association est une classe et à ce titre elle peut être mobilisée dans d'autres associations ou dans des héritages. Nous déconseillons néanmoins ces notations qui ont tendance à complexifier la lecture et la transformation du diagramme.

Nous conseillons donc de ne jamais associer une classe d'association.

Questions de synthèse

A quoi sert le LMD ?

.....
.....
.....
.....
.....
.....
.....

Pourquoi SQL n'est-il pas un langage de programmation ?

.....
.....
.....
.....
.....
.....
.....

Quel rapport y-a-t il entre le SQL et l'algèbre relationnelle ?

.....
.....
.....
.....
.....
.....
.....

Solutions des exercices

> **Solution** n°1

Exercice p. 5

```
1 SELECT *
2 FROM CLIENTS ;
```

> **Solution** n°2

Exercice p. 5

```
1 SELECT NP, NOMP
2 FROM PRODUITS
3 WHERE COUL='Rouge'
4 AND PDS>2000 ;
```

Complément

Si on ne sait pas comment la couleur a été saisie (minuscules/majuscules), il faut utiliser les fonctions de conversion de caractères telles que UPPER ou LOWER.

```
1 SELECT NP, NOMP
2 FROM PRODUITS
3 WHERE LOWER(COUL)='rouge'
4 AND PDS>2000 ;
```

> **Solution** n°3

Exercice p. 5

```
1 SELECT NR
2 FROM VENTES ;
```

Les vendeurs représentés dans la table ventes ont vendu quelque chose.

```
1 SELECT DISTINCT NR
2 FROM VENTES
3 ORDER BY NR ;
```

On peut dédoublonner et trier le résultat.

> **Solution** n°4

Exercice p. 5

```
1 SELECT DISTINCT C.NOMC
2 FROM CLIENTS C, VENTES V
3 WHERE V.NC=C.NC
4 AND V.QT>180
```

```
5 AND C.VILLE='Lyon' ;
```

> **Solution** n°5

Exercice p. 5

```
1 SELECT DISTINCT R.NOMR, C.NOMC
2 FROM REPRESENTANTS R, CLIENTS C, PRODUITS P, VENTES V
3 WHERE V.NC=C.NC
4 AND V.NR=R.NR
5 AND V.NP=P.NP
6 AND V.QT>100
7 AND P.COUL='Rouge' ;
```

> **Solution** n°6

Exercice p. 14

Que renvoie la dernière instruction SQL de la séquence ci-dessous ?

```
1 CREATE TABLE R (a INTEGER, b INTEGER);
2 INSERT INTO R VALUES (1,1);
3 INSERT INTO R VALUES (1,2);
4 INSERT INTO R VALUES (3,3);
5 SELECT * FROM R R1, R R2 WHERE R1.a=R2.a;
```

☐

1	1
1	2
3	3

☐

1	1
1	1
1	2
1	2
3	3
3	3

☐

1	1	1	1
1	1	1	2

1	1	3	3
1	2	1	1
1	2	1	2
1	2	3	3
3	3	1	1
3	3	1	2
3	3	3	3

☑

1	1	1	1
1	1	1	2
1	2	1	1
1	2	1	2
3	3	3	3

☐

1	1	1	2
1	1	3	3
1	2	1	1
1	2	3	3
3	3	1	1
3	3	1	2

> **Solution** n°7

Exercice p. 18

Soit les trois relations instanciées suivantes :

Numero	Nom	Prenom
1	Jacques	Brel
2	Georges	Brassens
3	Léo	Ferré

Relation Personne

Personne	Photo
1	1
2	1
3	1

Relation PersonnesPhotos

Numero	Date	Lieu
1	10/12/1965	Paris
2	18/03/2002	Lille

Relation Photo

Combien de tuples renvoie l'instruction SQL suivante :

```
SELECT *
FROM Photo ph LEFT JOIN (
 PersonnesPhotos pp INNER JOIN Personne pe
  ON pp.Personne=pe.Numero
 ) ON ph.Numero=pp.Photo;
```

4

"PersonnesPhotos INNER JOIN Personne" renvoie 3 tuples à 6 attributs et inclut les informations comprises dans PersonnesPhotos. La requête est donc équivalente à "SELECT Photo ph LEFT JOIN PersonnesPhotos pp ON ph. Numero=pp.Photo".

Jointure externe gauche donc :

- 1 tuple pour chaque jointure réalisée, donc photo numéro 1 avec chacun des 3 tuples de pp = *3 tuples*
- 1 tuple pour chaque tuple de ph non concerné par la jointure. Donc *1* tuple pour la photo numéro 2.

Soit 4 tuples au total

> **Solution** n°8

Exercice p. 20

```
R1=Restriction(PAIEMENT_LOYER,code_loc=X)
R2=Projection(R1,prix_paye)
```

```
SELECT prix_paye
FROM PAIEMENT_LOYER
WHERE code_loc='X'
```

> **Solution** n°9

Exercice p. 20

```
R1=Restriction(LOCATION,Code_loc=X)
```

```
R2=Jointure(R1,APPARTEMENT, R1.code_appt=APPARTEMENT.code_appt)
R3=Projection(R2,adresse)
```

```
SELECT adresse
FROM LOCATION l JOIN APPARTEMENT a
ON l.code_appt=a.code_appt
WHERE l.code_loc='X'
```

> **Solution** n°10

Exercice p. 20

```
R1=JointureExterneGauche(APPARTEMENT,LOCATION, APPARTEMENT.code_appt=LOCATION.
code_appt)
R2=Restriction (R1, code_loc=NULL)
```

```
SELECT *
FROM APPARTEMENT a LEFT JOIN LOCATION l
ON l.code_appt=a.code_appt
WHERE l.code_loc IS NULL
```

> **Solution** n°11

Exercice p. 20

```
SELECT Employe.Nom
FROM Employe JOIN Societe
ON  Employe.Societe=Societe.Nom
WHERE Employe.Fonction='Directeur' AND Societe.Pays='France'
```

> **Solution** n°12

Exercice p. 20

```
SELECT Num, Nom, Salaire*6.55957 As SalaireEnFrancs
FROM Employe
```

> **Solution** n°13

Exercice p. 21

```
SELECT Employe.Salaire, Fonction.SalaireMin, Fonction.SalaireMax, (Employe.
Salaire >= Fonction.SalaireMin AND Employe.Salaire <= SalaireMax) AS Correspond
FROM Employe JOIN Fonction
ON Employe.Fonction=Fonction.Intitule
```

> **Solution** n°14

Exercice p. 21

```
UPDATE Fonction
SET NbHeures=35, SalaireMin=SalaireMin*35/NbHeures, SalaireMax=SalaireMax*35
/NbHeures
WHERE NbHeures>35
```

Abréviations

SGBDR : Système de Gestion de Bases de Données Relationnelles

SQL : Structured Query Language