Modèle relationnel et langage SQL

C. MULLAERT

Lycée Saint-Louis

Notion de Relation et de Schéma

Définition : On utilise le terme de **Domaine** pour désigner un ensemble de valeurs.

Exemple :

- type numérique `Int` : $[\![-2^{31}, 2^{31}-1]\!]$, `Float` (décimal simple précision), `Double` (décimal double précision)
- chaîne de caractères de taille fixe `Char(n)` ou variable `Varchar(n)`.
- Date : `Date` et `Datetime`

Notion de Relation et de Schéma

Définition : Soit n un entier strictement positif et $D_1,...,D_n$ une famille finie de domaines. Une **relation** est la donnée d'un sous-ensemble fini du produit $D_1 \times ... \times D_n$.

C'est donc la donnée d'un nombre fini de n-uplets $(v_1, ..., v_n)$ tels que, pour tout $i \in [\![1, n]\!]$, on ait $v_i \in D_i$.

Remarque : Par définition, il est exclu que deux n-uplets appartenant à une relation R soient identiques.
Il n'y a pas d'ordre a priori sur les n-uplets d'une relation

Notion de Relation et de Schéma

Exemple : On prend $n = 2$, $D_1 =$`Int`, $D2 =$`Varchar(20)` et on définit une relation R par

$$R = \{(1, "\text{Aline}"), (2, "\text{Roméo}"), (3, "\text{Juliette}")\}$$

De manière plus commode, on représente une relation par un tableau:

R	Numéro	Nom
	1	Aline
	2	Romeo
	3	Juliette

Notion de Relation et de Schéma

Définition : Les chaines "Numéro" et "Nom" sont appelées **attributs** de la table. Chaque attribut est associé à un domaine.

Définition : Le **schéma** d'une relation est la liste de ses attributs.

Définition : Les éléments d'une relation R sont appelés **enregistrements**.

Opérations sur les relations

Définition : Soient R et S deux relations de même schéma $A = \{A_1, ..., A_n\}$. On définit de manière naturelle **l'union**, **la différence** et **l'intersection** de deux relations par :

$$(v_1, .., v_n) \in R \cup S \Leftrightarrow (v_1, .., v_n) \in R \text{ ou } (v_1, .., v_n) \in S$$

$$(v_1, .., v_n) \in R \setminus S \Leftrightarrow (v_1, .., v_n) \in R \text{ et } (v_1, .., v_n) \notin S$$

$$(v_1, .., v_n) \in R \cap S \Leftrightarrow (v_1, .., v_n) \in R \text{ et } (v_1, .., v_n) \in S$$

Exemples

Décrire les relations $R \cup S$, $R \cap S$ et $R \setminus S$ dans le cas suivant :

R	A_1	A_2	A_3
	1	1	0
	2	1	1
	1	1	3
	3	0	2

S	A_1	A_2	A_3
	1	1	0
	3	1	1
	1	1	3

Opérations sur les relations : le produit cartésien

Définition : Soient R et S deux relations de schémas $A = \{A_1, ..., A_n\}$ et $B = \{B_1, ..., B_p\}$. On suppose que leurs attributs sont différents.
Le **produit cartésien** $R \times S$ est la relation de schéma $A \cup B$ telle que

$$(v_1, ..., v_{n+p}) \in R \times S \Leftrightarrow (v_1, ..., v_n) \in R \text{ et } (v_{n+1}, ..., v_{n+p}) \in S$$

Exemples

Décrire le produit $R \times S$ dans le cas suivant :

R	A_1	A_2	A_3
	1	1	0
	2	1	1
	1	1	3

S	B_1	B_2
	1	1
	3	1

Opérations sur les relations : la projection

Définition : Soit R une relation et $(A_1, ..., A_n)$ la liste de ses attributs. On se donne une liste d'entiers $(i_1, ..., i_p)$ telle que $1 \leq i_1 < ... < i_p \leq n$.

On appelle **projection** de R sur $(A_{i_1}, .., A_{i_p})$ et on note $\pi_{(A_{i_1},..,A_{i_p})}(R)$, la nouvelle relation R' de schéma $(A_{i_1}, .., A_{i_p})$ et telle que

$$(v_1, ..., v_p) \in R' \Leftrightarrow \exists (x_1, ..., x_n) \in R \mid \forall j \in [\![1, p]\!], v_j = x_{i_j}$$

Plus simplement, on ne garde que certains attributs.

Opérations sur les relations : la sélection

Définition : Une **condition de sélection** est une fonction du produit $D_1 \times ... \times D_n$ dans l'ensemble $\{\texttt{True},\texttt{False}\}$. Elle associe donc à chaque enregistrement $(v_1, ..., v_n)$ un élément de l'ensemble $\{\texttt{True},\texttt{False}\}$.

Définition : Soit R une relation et F une condition de sélection. On appelle **sélection** de R par F et on note $\sigma_F(R)$, la nouvelle relation R' de même schéma que R et définie par

$$R' = R \cap F^{-1}(\{\texttt{True}\})$$

Plus simplement, on ne garde que les enregistrements pour lesquels la condition de sélection est vérifiée.

Exemples

Décrire la projection $\pi_{\text{Note}}(R)$ et la sélection $\sigma_{\text{Note}>10}(R)$ dans le cas suivant :

R	Nom	Note
	Roméo	12
	Juliette	8
	Aline	15

Opérations sur les relations : la jointure naturelle

Définition : Soient R et S deux relations de schémas $A = \{A_1, ..., A_n\}$ et $B = \{B_1, ..., B_p\}$.

La **jointure naturelle** $R \bowtie S$ est la relation de schéma $A \cup B$ définie par

$$\boldsymbol{x} \in R \bowtie S \Leftrightarrow \pi_A(\{\boldsymbol{x}\}) \in R \text{ et } \pi_B(\{\boldsymbol{x}\}) \in S$$

Plus simplement, les enregistrements de $R \bowtie S$ sont ceux dont les attributs communs aux deux relations $(A \cap B)$ coïncident.

Exemples

Décrire la jointure naturelle $R \bowtie S$ dans le cas suivant :

R	Nom	Note
	Roméo	12
	Juliette	8
	Aline	15
	Roméo	7

S	Nom	Commentaire
	Roméo	Bien
	Juliette	Encourageant
	Boris	En progression

Définition : Une **clé candidate** est un ensemble d'attributs tel qu'il n'existe pas deux enregistrements qui ont les mêmes valeurs pour ces attributs

Définition : Une **clé primaire** est un choix particulier de clé candidate

Définition : Une **clé étrangère** est un attribut (ou un ensemble d'attributs) destiné à contenir uniquement des valeurs prises par la clé primaire d'une autre table.

Exemple : Une bibliothèque dispose de trois tables :

- Une table `lecteurs` avec comme attributs un numéro unique d'identification, le nom et le prénom
- Une table `livres` avec comme attributs un numéro unique, le titre du livre et sa côte.
- Une table `prêt` avec comme attributs le numéro du livre emprunté, le numéro du lecteur concerné ainsi que la date de retour prévu.

Identifier les clés candidates de chaque table. Proposer des clés primaires et étrangères.

Entités et associations

Lorsque l'on conçoit une base de données on définit des **entités** et les **associations** entre elles qui les relient naturellement.

L'association entre `lecteurs` et `prêts` est du type "un à plusieurs" que l'on note 1-$*$. Elle utilise le principe de clé étrangère.

L'association entre `lecteurs` et `livres` est du type "plusieurs à plusieurs" que l'on note $*$-$*$. La table `prêts` permet de la séparer en deux associations 1-$*$.

Il y a également des relations "1-1" par exemple entre des tables `pays` et `capitale` (mais on pourrait alors fusionner les tables).

L'instruction de base : `SELECT` permet de réaliser

- Projection
- Selection
- Jointure

Exemple : La projection sur tous les attributs de la table city :

```
SELECT * FROM city
```

Exemple : Les noms et population des villes de la table city :

```
SELECT Name, Population FROM city
```

Pour réaliser une sélection, on utilise la clause `WHERE`

Exemple : Les villes de la table city dont la population est supérieure à 1000000:

```
SELECT Name FROM city WHERE Population>1000000
```

Exemple : Les pays de la table country situées en Europe:

```
SELECT Name FROM country WHERE Continent='Europe'
```

Pour calculer des agrégats, on utilise des fonctions d'agrégations :

- MIN(Population), MAX(Population) renvoie la population minimale et maximale.
- SUM(Population) renvoie la somme des populations
- AVG(Population) renvoie la population moyenne
- COUNT(Continent) renvoie le nombre de continents

Exemple : Le nombre d'habitant du pays d'Europe le plus peuplé :

```
SELECT MAX(Population) FROM country WHERE
Continent='Europe'
```

Exemple : Le nombre de pays de la table country situés en Europe:

```
SELECT COUNT(*) FROM country
WHERE Continent='Europe'
```

Les résultats de la requête peuvent être groupés par paquets à l'aide de la clause `GROUP BY` :

Exemple : Pour chaque continent, le nombre de pays dont la population est supérieure à 60000000 habitants:

```
SELECT Continent, COUNT(*) FROM country
WHERE Population>60000000 GROUP BY Continent
```

Remarque : L'intérêt de la clause `GROUP BY` réside dans l'utilisation des fonctions d'agrégation

Remarque : Attention, la clause `WHERE` sélectionne ici les enregistrements avant l'agrégation

Si l'on veut éviter les doublons, on utilise DISTINCT

Exemple : La liste des noms des villes présentes dans city sans doublons

```
SELECT DISTINCT Name FROM city
```

Exemple : Le nombre de noms différents de villes présentes dans city

```
SELECT COUNT(DISTINCT Name) FROM city
```

On peut appliquer une condition de sélection sur les groupes à l'aide de la clause `HAVING`

Exemple : La liste des continents qui comportent plus de 50 pays

```
SELECT Continent FROM country
GROUP BY Continent HAVING COUNT(*)>50
```

Remarque : Attention, la clause `WHERE` sélectionne les enregistrements et la clause `HAVING` sélectionne des groupes

On peut ordonner les résultats selon un attribut à l'aide de la clause `ORDER BY` suivie de `ASC` (par défaut) ou `DESC`

On peut limiter le nombres d'enregistrement avec `LIMIT` et/ou utiliser `OFFSET` pour ne faire apparaître les enregistrements qu'à partir d'un certain rang.

Exemple : La liste des pays par ordre de population décroissante

```
SELECT Name FROM country ORDER By Population DESC
```

Exemple : La liste des 10 villes les moins peuplées avec leur nombre d'habitants

```
SELECT Name, Population FROM city
ORDER By Population DESC LIMIT 10
```

Exemple : La liste des 20 suivantes

```
SELECT Name, Population FROM city
ORDER By Population LIMIT 20 OFFSET 10
```

Pour réaliser une jointure, on peut réaliser une sélection sur le produit cartésien :

Exemple : La liste des villes avec leur pays d'appartenance

```
SELECT city.Name, country.Name FROM city , country
                WHERE CountryCode = Code
```

Remarque : On a précisé la table de l'attribut Name pour lever l'ambiguïté entre les attributs présents dans les deux tables. Pour cela, on peut également utiliser des alias et ce sera indispensable lors d'une auto-jointure.

Exemple : La liste des villes avec leur pays d'appartenance

```
SELECT Ci.Name,Co.Name FROM city AS Ci, country AS Co
                WHERE CountryCode = Code
```

Autre possibilité : on utilise les mots clés `JOIN ... ON`.

```
SELECT Ci.Name, Co.Name FROM city AS Ci
JOIN country AS Co ON Ci.CountryCode = Co.Code
```

ou

```
SELECT Ci.Name, Co.Name FROM city AS Ci
JOIN country AS Co ON CountryCode = Code
```

Remarque : On privilégiera ces écritures car elle permet de séparer les critères de jointures (`ON ...`) des critères de sélection (`WHERE...`)

Pour obtenir la liste des codes des pays où l'anglais et le français sont parlés, la requête

```
SELECT Code from countrylanguage
WHERE Language="French" AND Language="English"
```

ne renvoie rien mais on peut faire une auto-jointure

```
SELECT L1.CountryCode
FROM countrylanguage AS L1 JOIN countrylanguage AS L2
ON L1.countrycode=L2.countrycode
WHERE L1.language="French" AND L2.language="English"
```

La requête suivante renvoie une valeur unique

```
SELECT AVG(Population) FROM city
```

Elle peut devenir une **sous-requête** si on la place entre parenthèses

```
SELECT Name FROM city
WHERE Population>=(SELECT AVG(Population) FROM city)
```

Exemple : La liste des codes des pays où l'anglais et le français sont parlés

```
SELECT CountryCode FROM countrylanguage
WHERE language="English" and Countrycode in
(SELECT CountryCode FROM countrylanguage WHERE
language="French" )
```

On peut réaliser une union/ intersection/ exclusion de requête (même si souvent on peut s'en sortir avec AND/OR/AND NOT) en utilisant UNION/INTERSECT/EXCEPT.

Exemple : La liste des noms de pays où le français est parlé mais pas l'anglais (selon la base de données)

```
(SELECT Name from country JOIN countrylanguage ON
CountryCode=Code WHERE Language="French")
EXCEPT
(SELECT Name from country JOIN countrylanguage ON
CountryCode=Code WHERE Language="English")
```

Exemple : La liste des noms des pays où l'anglais et le français sont parlés :

```
(SELECT Name from country JOIN countrylanguage ON
CountryCode=Code WHERE Language="French")
INTERSECT
(SELECT Name from country JOIN countrylanguage ON
CountryCode=Code WHERE Language="English")
```

Plus lisible que

```
SELECT Name
FROM countrylanguage AS L1
JOIN countrylanguage AS L2 ON
L1.countrycode=L2.countrycode
JOIN country ON L1.countrycode=code
WHERE L1.language="French" AND L2.language="English"
```

Récapitulatif : Les mots clefs sont toujours dans l'ordre suivant (s'ils apparaissent) dans une requête :

`SELECT...FROM...WHERE...GROUP BY...HAVING...ORDER BY...LIMIT...OFFSET...`