Introduction aux bases de données et au langage SQL

Ce document ne prétend en aucun cas remplacer les ouvrages ou manuels de référence disponibles en librairie ou sur Internet. Il s'agit simplement d'un aide-mémoire permettant une prise en main rapide et présentant les concepts généraux utiles.

Le manuel de référence de MySQL est disponible en ligne à cette adresse :
http://dev.mysql.com/doc/refman/5.5/en/index.html

De très nombreux cours sur les bases de données relationnelles, en français, se trouvent très facilement sur la toile.

Introduction

Le responsable d'une bibliothèque peut être tenté de gérer les emprunts par un simple fichier sur un tableur. Les premières lignes de ce fichier peuvent ressembler à ce qui suit.

index	titre	genre	auteurN	auteurP	nom	prénom	tél	date	retour
1	Salambo	Roman	Flaubert	Gustave	Castel	Claude	06123451 12	12-août-10	23-août-10
2	93	Roman	Hugo	Victor	Le Dray	Camille	0422113827	17-sept.-10	03-oct.-10
3	Mme Bovary	Littérature	Flaubert	Gustave	Loutard	Annie	064331282	01-nov.-10	02-déc.-10
1	Salambô	Roman	Flaubert	Gustave	Filatre	Jean	0432168719	02-sept.-10	14-sept.-10
4	Aphorismes	Littérature	Wilde	Oscar	Castel	Marie	0412324494		
5	L'immoraliste	Roman	Gide	André	Biraud	Michèle	0434578612	05-févr.-11	
3	Madame Bovary	Roman	Flaubert	Gustave	Castel	Claude	0412324494	06-janv.-10	
2	Quatre-vingt-treize	Roman	Hugo	Victor	Filâtre	Jean	0432168719	20-oct.-10	03-nov.-10
1	Salambo	Littérature	Flaubert	gustave	Ledray	Camille	0422113627	16-sept.-10	27-sept.-10
…	…	…	…	…	…	…	…	…	…

Toutefois, de nombreuses difficultés se font très vite jour : l'orthographe des différents noms de famille peut manquer de cohérence (Le Dray ou Ledray), l'écriture des titres des ouvrages présente des problèmes analogues (Mme ou Madame Bovary), la recopie des numéros de téléphone peut être fastidieuse et donc erronée, et même l'entrée de la date peut poser problème (le deuxième emprunt de Mme Bovary date certainement du 6 janvier 2011 et non 2010).

L'exemple ci-dessus est évidemment volontairement de taille très réduite. Mais on imagine facilement que, dans une situation réelle, le fichier produit puisse contenir des milliers de lignes. Il devient alors très coûteux (en temps de calcul) de retrouver les prêts de M. Castel, ou encore de

chercher quels sont les ouvrages qui ont été prêtés depuis plus de 2 semaines. La conception « plate » du fichier de gestion de la bibliothèque impose une lecture séquentielle complète du tableau pour chaque interrogation.

L'invention du modèle relationnel, fondé sur la définition de ce qu'on appelle aujourd'hui l'algèbre relationnelle, a permis de résoudre ces problèmes d'intégrité et de cohérence des données tout en assurant l'efficacité des recherches et interrogations des bases de données.

C'est ainsi que les systèmes de gestion de bases de données (SGBD) actuels permettent l'utilisation de quantités considérables de données dans des bases partagées par de multiples utilisateurs et souvent disponibles en ligne.

C'est ce modèle relationnel que nous allons présenter ici.

5 tables au lieu d'un fichier tableur

Nous créons dans cet exemple 5 tables, qui contiennent l'ensemble de l'information utile, mais sous une forme beaucoup plus utilisable, comme nous allons le voir.

Livres			
id	auteur	titre	genre
1	1	Salambô	1
2	2	Quatre-vingt-treize	1
3	3	L'immoraliste	1
4	4	Aphorismes	2
5	1	Madame Bovary	1
…	…	…	…

La table des Livres comporte une colonne identifiant le livre, puis une autre identifiant l'auteur par son numéro, et de même pour le genre. Les titres ne sont pas répétés, ce qui garantit la cohérence des données.

Auteurs		
id	nom	prénom
1	Flaubert	Gustave
2	Hugo	Victor
3	Gide	André
4	Wilde	Oscar
…	…	…

Les mêmes remarques s'appliquent à la table des Auteurs.

Emprunts			
qui	quoi	date	rendu
1	1	12-août-10	23-août-10
6	1	02-sept.-10	14-sept.-10
2	1	16-sept.-10	27-sept.-10
2	2	17-sept.-10	03-oct.-10
6	2	20-oct.-10	03-nov.-10
3	5	01-nov.-10	03-déc.-10
1	5	06-janv.-11	*NULL*
5	3	05-févr.-11	*NULL*
…	…	…	…

La table des Emprunts permet de trouver le numéro de l'emprunteur et celui de l'ouvrage emprunté. La date de rendu est éventuellement NULL si l'ouvrage est encore dehors.

Emprunteurs			
id	nom	prénom	tél
1	Castel	Claude	0612345112
2	Le Dray	Camille	0422113827
3	Loutard	Annie	0654331282
4	Castel	Marie	0412324494
5	Biraud	Michèle	0434578612
6	Filâtre	Jean	0432168719
…	…	…	…

Là encore, la non répétition des abonnés de la bibliothèque permet de garantir la cohérence des données.

Genre	
id	genre
1	Roman
2	Littérature
…	…

Vocabulaire du modèle relationnel

Les noms des colonnes de chaque table s'appellent les *attributs*. Quand on formalise une table quelconque, on note en général les attributs A_1, A_2, A_3, A_4… À chaque attribut est associé son domaine, qui est l'ensemble des valeurs qu'il peut prendre : un entier, une chaîne de caractères, etc. On note $\mathrm{Dom}(A_i)$ le domaine de l'attribut A_i.

Chaque ligne s'appelle un *n-uplet* (ou *tuple* en anglais). C'est donc un élément de $\mathrm{Dom}(A_1)\times\mathrm{Dom}(A_2)\times\mathrm{Dom}(A_3)\times\ldots$ L'ensemble des n-uplets de la table s'appelle une *relation*, c'est donc une partie de $\mathrm{Dom}(A_1)\times\mathrm{Dom}(A_2)\times\mathrm{Dom}(A_3)\times\ldots$

La base de données est ainsi constituée d'un nombre fini de relations : une par table.

On dit que la relation r suit le *schéma de relation* $R(A_1,A_2,A_3,\ldots)$.

L'ensemble des schémas de relation s'appelle le *schéma de la base de données*.

Notation : si t est le n-uplet (3, Loutard, Annie, 0654331282), on note t[id,prénom] = (3, Annie), t[nom]=(Loutard) et ainsi de suite.

Notions de clé

On appelle *super-clé* d'une relation r un ensemble K d'attributs tel que

$$\forall t, u \in r, t[K] = u[K] \Rightarrow t = u.$$

Une *clé* (ou encore *clé candidate*) d'une relation r est alors simplement une super-clé de taille minimale (pour l'inclusion).

Une *clé primaire* est simplement un choix d'une clé candidate. Pour l'indiquer, on souligne les attributs correspondants dans la table. On la nomme souvent id, ce qui a été fait dans quatre des cinq tables ci-dessus. Pour la table Emprunts, c'est le triplet qui, quoi, date qui peut servir de clé primaire.

Indiquer au système une clé primaire pour chaque table permet une indexation des données à l'aide de cette clé, ce qui renforce l'efficacité des procédures d'interrogation de la table.

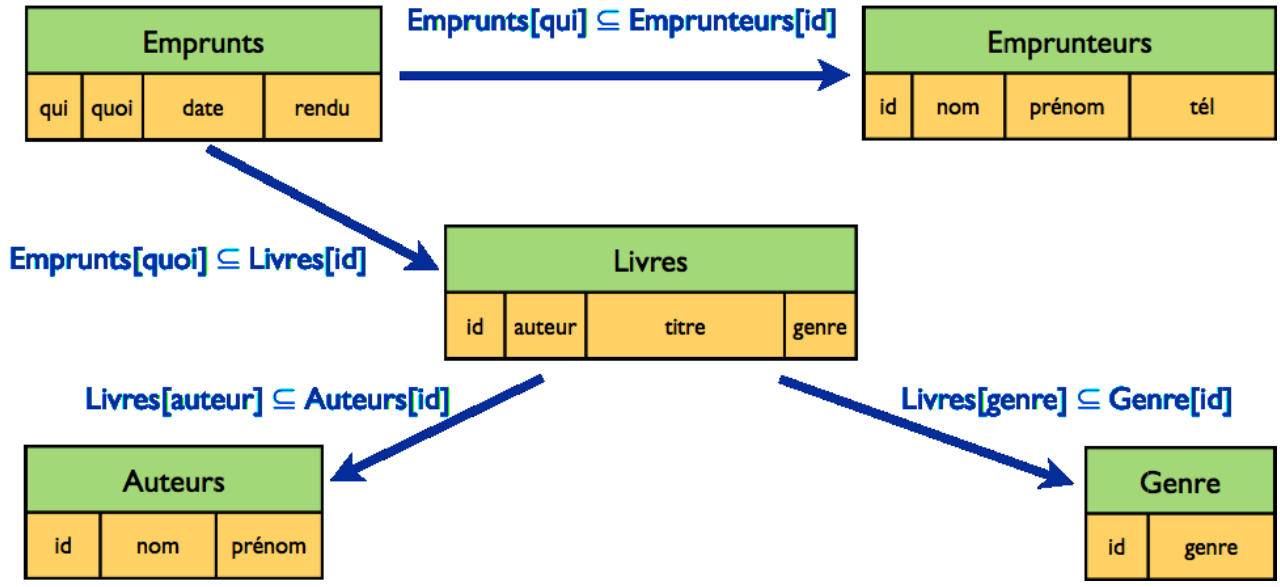
Notion de clé étrangère (foreign key)

Considérons la table Emprunts : elle possède un attribut quoi, qui fait le lien avec l'attribut id de la table Livres. C'est ce qu'on appelle une *clé étrangère*.

Cela permet d'assurer :

- qu'on ne peut pas insérer une ligne dans la table des Emprunts avec une valeur de l'attribut quoi qui n'existe pas dans la table des Livres ;
- qu'on ne peut pas supprimer une ligne de la table des Livres si au moins une ligne de la table des Emprunts a une valeur de l'attribut quoi qui corresponde à la ligne à supprimer.

Voici le graphe des clés étrangères pour nos 5 tables.



Notons que ce graphe permet de trouver un ordre logique de création des tables : on commencera par créer les tables Emprunteurs, Auteurs et Genre (d'où ne sortent aucune clé étrangère), puis la table Livres, et on terminera par la table Emprunts.

Opérateurs de l'algèbre relationnelle

Pour deux relations qui ont le même schéma, on peut bien entendu utiliser les opérateurs ensemblistes usuels d'union, intersection et différence.

Mais on peut également définir de façon naturelle un produit cartésien. Si r a pour schéma R et s a pour schéma S, alors :

$$r \times s = \{t, t[R] \in r, t[S] \in s\}.$$

Opérateurs spécifiques de l'algèbre relationnelle

Nous en venons maintenant aux opérateurs spécifiques de l'algèbre relationnelle.

La sélection (ou retriction), notée en général σ

Une formule de sélection F est une formule construite à partir des constantes, des attributs, des fonctions usuelles, des opérateurs de comparaison, des connecteurs logiques (et, ou, non).

Par exemple : nom= « Loutard » ET date < 1/1/2011.

Étant donnée une formule de sélection F, on définit alors :

$$\sigma_F(r) = \{t \in r, t \text{ satisfait } F\}.$$

Il s'agit donc de sélectionner les n-uplets qui satisfont telle ou telle condition.

La projection, notée en général π

Il s'agit ici de récupérer un sous-ensemble des composantes des n-uplets de la relation considérée.

Plus formellement, si X est un sous-ensemble d'attributs du schéma d'une relation r, on a donc :

$$\pi_X(r) = \{t[X], t \in r\}.$$

Le renommage, noté en général ρ

Il s'agit tout simplement de renommer un attribut (cela pourra en particulier servir lors de jointures — cf. plus loin — pour utiliser deux tables dont certains attributs portent le même nom alors qu'ils recouvrent des entités différentes).

Si R est le schéma de la relation r, si A est un attribut du schéma, mais pas B, on peut renommer A en B :

$$rho_{A \to B}(r) = \{t, \exists u \in r, t[A] = u[B] \text{ et } t[C] = u[C] \text{ si } C \neq A\}.$$

La jointure, notée en général ⋈

Étant donnée une formule de sélection F et deux relations r et s de schéma respectifs R et S, on définit :

$$r \bowtie_F s = \{t, t[R] \in r, t[S] \in s, t \text{ satisfait } F\}.$$

On peut remarquer que l'on obtient le même résultat avec $\sigma_F(r \times s)$.

La division, notée en général ÷

C'est un peu l'opération inverse du produit cartésien : on considère une relation r de schéma $R(A_{1,} A_{2,} \ldots, A_m, A_{m+1}, \ldots, A_n)$ et une relation s de schéma $S(A_{m+1}, \ldots, A_n)$.

On définit alors :

$$r \div s = \{t, \forall t' \in s, (t, t') \in r\}.$$

L'agrégation, notée en général ɣ

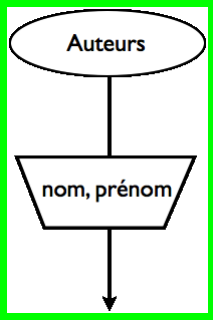
On considère un jeu X d'attributs d'une relation r. On rassemble les n-uplets t de r qui ont les mêmes valeurs sur les attributs de X : on obtient, pour chaque valeur commune t[X], un sous-ensemble de n-uplets. **Pour chacun de ces sous-ensembles,** on calcule $(f_1(A_{i_1}), f_2(A_{i_2}), f_3(A_{i_3}), \ldots)$ où les f_k sont des fonctions à choisir parmi AVG (moyenne), COUNT, SUM, MAX, MIN… et où les A_{i_k} sont des attributs qui figurent ou non dans X.

On verra des exemples plus bas : le langage SQL utilise les mots clefs GROUP BY et HAVING pour cette opération.

Le langage SQL

Le langage SQL permet de traduire les expressions de l'algèbre relationnelle, avec une syntaxe très simple et un jeu réduit de mots clefs.

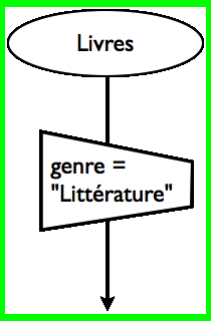
Projection

SQL	SELECT nom, prénom FROM Auteurs ;
Algèbre relationnelle	$\pi_{\text{nom, prénom}}(\text{Auteurs})$
Diagramme	

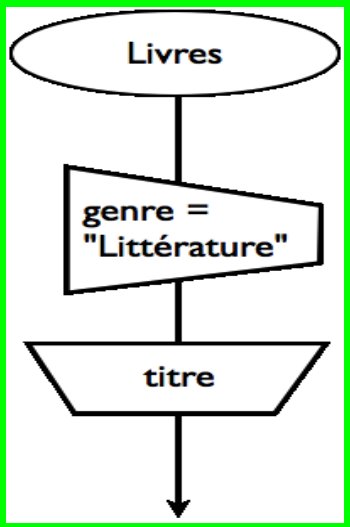
Alors que l'algèbre relationnelle travaille sur de vrais ensembles au sens mathématique, SQL peut renvoyer des *multi-ensembles* : c'est-à-dire que la même valeur peut être répétée. Pour éviter ce comportement, on utilise une variante de la syntaxe, à l'aide du mot clef DISTINCT.

SQL	SELECT DISTINCT nom FROM Emprunteurs ;
Algèbre relationnelle	$\pi_{\text{nom}}(\text{Emprunteurs})$

Sélection

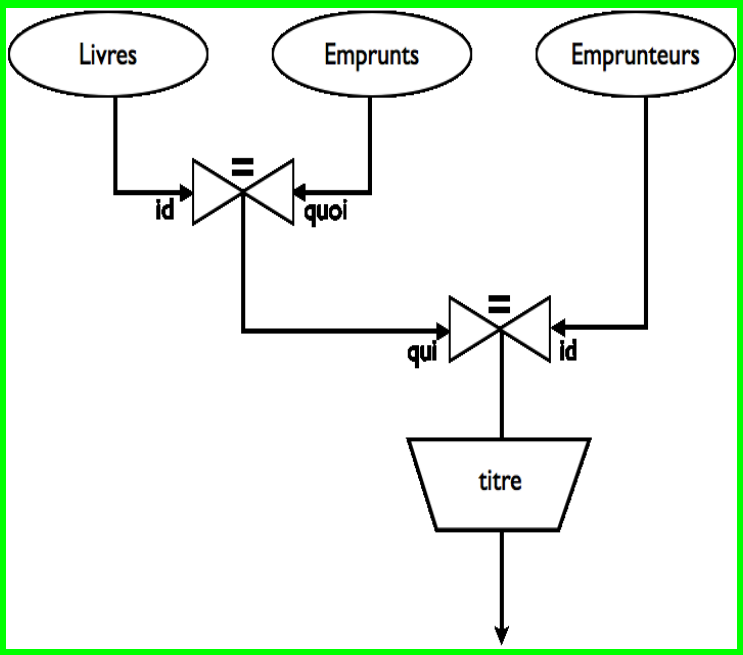
SQL	SELECT * FROM Livres WHERE genre = « Littérature »;
Algèbre relationnelle	$\sigma_{\text{genre} = \text{« Littérature »}}(\text{Livres})$
Diagramme	

Sélection et projection

SQL	SELECT titre FROM Livres WHERE genre = « Littérature »;
Algèbre relationnelle	$\sigma_{\text{genre} = \text{« Littérature »}}(\text{Livres})$
Diagramme	

Jointure

Voici un exemple plus compliqué, pour illustrer la jointure.

SQL	SELECT titre FROM (Livres JOIN Emprunts ON Livres.id = Emprunts.quoi) JOIN Emprunteurs ON Emprunteurs.id = qui WHERE Emprunteurs.nom = « Biraud »;
Diagramme	

Exercices

Quels sont les titres des livres empruntés par Mme Loutard ?

```
SELECT
   Livres.titre
FROM
   (Livres JOIN Emprunts
      ON Livres.id = Emprunts.quoi)
   JOIN
   Emprunteurs
      ON Emprunts.qui = Emprunteurs.id
WHERE
   Emprunteurs.nom LIKE "Loutard" ;
```

Qui a emprunté au moins deux livres ?

```
SELECT
   nom, prénom
FROM
   Emprunts JOIN Emprunteurs
      ON Emprunts.qui = Emprunteurs.id
GROUP BY
  nom, prénom
HAVING
  COUNT(*) > 1
ORDER BY
  nom, prénom ;
```

La clause `ORDER BY` permet de rendre les résultats rangés en ordre alphabétique, d'abord des noms, puis des prénoms.

La clause `GROUP BY` permet de préparer l'agrégation en spécifiant les critères de définition des sous-ensembles : on regroupe ici par personne concernée. `HAVING` fait un peu la même chose que `WHERE`, mais permet d'utiliser les fonctions d'agrégation, en l'occurence ici `COUNT` qui compte le nombre d'éléments de chaque sous-ensemble (donc le nombre de prêts par personne, ici).

Qui a encore un livre emprunté non rendu ?De quel livre s'agit-il ?

```
SELECT
   E.nom, E.prénom, L.titre, A.nom as "auteur"
FROM
   (Emprunts as P JOIN Emprunteurs as E
      ON P.qui = E.id)
   JOIN
   (Livres as L JOIN Auteurs as A ON L.auteur = A.id)
      ON P.quoi = L.id
WHERE
   P.rendu is NULL ;
```

Remarque : on ne peut pas écrire `= NULL`, il faut utiliser `is NULL`, à cause de conventions particulières sur la constante `NULL`.

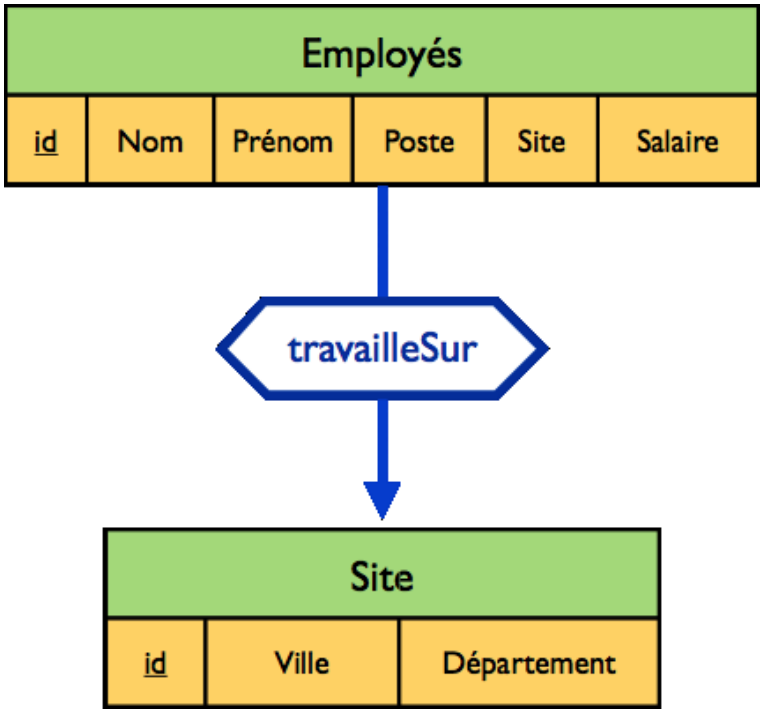
Le mot clef `AS` permet de renommer un attribut pour préparer l'affichage du résultat, mais aussi de donner un synonyme à une table, pour alléger l'écriture du code SQL. Les mots clefs peuvent être écrits en majuscules comme en minuscules.

Qui a encore un livre emprunté non rendu depuis au moins deux semaines ?De quels livres s'agit-il ?

```
SELECT
   E.nom, E.prénom, L.titre, A.nom as "auteur",
   DATEDIFF(NOW(), P.date) as "durée"
FROM
   (Emprunts as P JOIN Emprunteurs as E
      ON P.qui = E.id)
   JOIN
   (Livres as L JOIN Auteurs as A ON L.auteur = A.id)
      ON P.quoi = L.id
WHERE
   DATEDIFF(NOW(),P.date) >  14 AND P.RENDU is NULL
ORDER BY DATEDIFF(NOW(),P.date) DESC ;
```

La fonction `NOW` renvoie la date d'aujourdh'ui. La fonction `DATEDIFF` permet de calculer le nombre de jours entre deux dates. Le mot clef `DESC` permet d'avoir un ordre inverse (décroissant).

Complément : sélections emboîtées



Quels sont les employés sur le même site que Martin ?

```
SELECT
   Nom, Prénom
FROM
   Employés
WHERE
   Site = ( SELECT Site FROM Employés WHERE Nom = "Martin" ) ;
```

C'est correct car le SELECT interne renvoie une seule valeur.

Quels sont les employés ayant le même poste que Faroux et un salaire au moins égal à celui de Barcolle ?

```
SELECT
   Nom, Prénom, Salaire
FROM
   Employés
WHERE
   Poste = ( SELECT Poste FROM Employés WHERE Nom = "Faroux" )
   AND Salaire >= (SELECT Salaire FROM Employés
                     WHERE Nom = "Barcolle" ;
```

C'est correct car chaque `SELECT` interne renvoie une seule valeur.

Quels sont les employés gagnant plus que tous les employés des Bouches-du-Rhône ?

```
SELECT
   Nom, Prénom, Salaire
FROM
   Employés
WHERE
   Salaire > ALL
     ( SELECT Salaire
       FROM Employés E JOIN Sites S ON E.Site = S.id
       WHERE Département = 13 ) ;
```

Il existe aussi bien `> ALL`, `> ANY`.

`= ANY (SELECT …)` est équivalent à `IN (SELECT …)`

`!= ALL (SELECT …)` est équivalent à `NOT IN (SELECT ...)`

Mises à jour

Ce document est mis à jour régulièrement. La dernière version est disponible à cette adresse, où on trouvera également d'autres documents utiles : http://goo.gl/AVV5t

Auteur

Laurent Chéno, laurent.cheno@education.gouv.fr

Licence d'utilisation de ce document