I Le langage SQL

1. Définitions

— Etant donné des ensembles E_1 , E_2 , .. E_n, on appelle **table** tout sous ensemble de $E_1 \times E_2 \times .. \times E_n$. Ainsi, une **table** est un ensemble de n-uplets. Une **base de données relationnelle** peut être vue comme un ensemble de tables.
— Un **enregistrement** de la table est l'un de ces n-uplets. Un enregistrement s'appelle aussi **relation**.
— Un **attribut** de la table est le nom qui désigne les éléments d'un même ensemble permettant de constituer la table.
— Le **domaine** d'un attribut est le type d'éléments de l'ensemble qu'il désigne.

On considérera par exemple une table `tableau` représentant la classification des éléments de Mendeleiev dont le schéma est le suivant :

nom	numéro_atomique	symbole	colonne	ligne	bloc
Hydrogène	1	H	1	1	s
Hélium	2	He	18	1	s
Lithium	3	Li	1	2	s
Béryllium	4	Be	2(IIA)	2	s

Un enregistrement de cette table est (Hydrogène,1,H,1,1,s) ; les attributs sont `nom`, `numéro_atomique`, `symbole`, `colonne`, `ligne` et `bloc` dont les domaines sont tous des chaînes de caractères sauf le `numéro_atomique` et la `ligne` qui sont des entiers (les numéros des colonnes ne sont pas que des entiers).
On appelle **clé primaire** d'une table un (ou plusieurs) attribut qui permet d'identifier de façon unique un enregistrement. Dans notre exemple, le `nom`, le `numéro_atomique` et le `symbole` sont des clés primaires.
Si aucun attribut ne permet d'identifier un enregistrement de façon unique, une clé primaire peut être constituée de deux attributs (par exemple le nom de l'auteur **et** le titre du livre peuvent identifier une oeuvre).
Dans le cas où l'on ne peut pas utiliser un des attributs comme clé primaire, on rajoute souvent un attribut (appelé par exemple `id`) qui est un simple numéro (on numérote tout simplement les lignes).

2. Requêtes de base

Une requête SQL se formule en utilisant les commandes suivantes :

```
SELECT  attributs           # attributs à chercher (* pour tous les attributs)
SELECT DISTINCT             # Même chose sans les doublons
FROM   tables               # Lieu de la recherche
WHERE  conditions           # Critère de sélection
ORDER BY expression         # Trier les résultats
LIMIT n                     # Limiter à n enregistrements
OFFSET n                    # Débuter à partir de n enregistrements
```

Une requête doit théoriquement se terminer par « `;` », mais beaucoup de logiciels permettant d'interroger une base de données tolèrent son absence.

Exemple(s) :

⋆ La requête :
`SELECT symbole,numéro_atomique FROM tableau WHERE ligne = 3 ORDER BY bloc` renvoie la table contenant les symboles et les numéros atomiques des éléments de la ligne 3 en les triant par bloc (dans l'ordre alphabétique).

⋆ Les opérations, comparaisons et opérateurs logiques obéissent à une syntaxe classique (`+,-,*,/,<, <=,>,>=,=,!=,AND, OR,...`). Par exemple la requête :
`SELECT nom, numéro_atomique/ligne AS numéro_réduit FROM tableau WHERE ligne=4 AND bloc !='d'` renvoie la table contenant les noms et la partie entière du rapport entre le numéro atomique et le numéro de la ligne (rebaptisé « `numéro_réduit` ») des éléments de la quatrième ligne qui ne sont pas dans le bloc d.

3. Fonctions d'agrégation, GROUP BY et HAVING

Les **fonctions d'agrégation** permettent d'exécuter une requête sur un groupe d'enregistrement. Il existe différentes fonctions d'agrégation, les suivantes sont celles qui permettent de faire des statistiques sur une table :

nom	Action
`COUNT()`	Compte le nombre d'enregistrements d'un attribut
`AVG()`	Calcule la moyenne d'un attribut d'un ensemble d'enregistrements de type numérique
`MIN()`	Calcule le minimum d'un attribut d'un ensemble d'enregistrements de type numérique
`MAX()`	Calcule le maximum d'un attribut d'un ensemble d'enregistrements de type numérique
`SUM()`	Calcule la somme d'un attribut d'un ensemble d'enregistrements de type numérique

Exemple(s) :

- ⋆ La requête `SELECT COUNT(*) FROM tableau` renvoie le nombre d'enregistrements de la table, ie le nombre de lignes.
- ⋆ `SELECT SUM(numéro_atomique) FROM tableau WHERE ligne = 2` renvoie 52, la somme des numéros atomiques des éléments de la deuxième ligne.

On peut également regrouper les attributs selon certains critères en utilisant `GROUP BY`, et rajouter des conditions sur les groupes avec `HAVING` (placé après `GROUP BY`).

Exemple(s) :

- ⋆ `SELECT SUM(numéro_atomique) FROM tableau GROUP BY ligne` renvoie la somme des numéros atomiques de chaque ligne.
- ⋆ `SELECT SUM(numéro_atomique) FROM tableau GROUP BY ligne HAVING COUNT(*) > 8` renvoie la somme des numéros atomiques de chaque ligne qui possède au moins 9 éléments (donc à partir de la ligne 4).

4. Jointure

L'intérêt principal des bases de données est de pouvoir manipuler plusieurs tables en même temps et de pouvoir en croiser les résultats. On utilisera les deux autres tables `découverte` et `grandeurs` dont les premières lignes sont les suivantes :

nom	pays	symbole	date
Henry Cavendish	Grande-Bretagne	H	1766
Jules Janssen	Grande-Bretagne	He	1895
Joseph Norman Lockyer	Grande-Bretagne	He	1895
Johan August Arfwedson	Suède	Li	1817

La table `découverte`

symbole	remplissage	masse_atomique	temp_fusion	temp_ébullition
H	1	1.00794		
He	2	4.002602		-268.93
Li	2|1	6.94100	180.5	1342
Be	2|2	9.012182	1287	2471

La table `grandeurs`

La jointure de deux tables se fait par l'utilisation de la commande `table1 JOIN table2 ON condition` ; la `condition` permet de faire la liaison entre les deux tables.

Exemple(s) :

- ⋆ `SELECT DISTINCT pays FROM découverte JOIN tableau ON découverte.symbole = tableau .symbole WHERE ligne = 2` renvoie les pays, sans répétition, où ont été découverts les éléments de la ligne 2. La jointure se fait par l'identification `découverte.symbole=tableau.symbole` ; l'utilisation des préfixes `découverte` et `tableau` est nécessaire car `symbole` est un attribut dans les deux tables. Par contre comme `ligne` et `pays` ne sont pas des attributs des deux tables, le préfixe est inutile.
- ⋆ Pour alléger les notations des préfixes, on peut utiliser des alias : `SELECT DISTINCT pays FROM découverte d JOIN tableau t ON d.symbole = t.symbole WHERE ligne = 2` a le même effet, on utilise l'alias `d` pour `découverte` et `t` pour `tableau`.
- ⋆ On peut alors exécuter des jointures multiples :

```
SELECT DISTINCT d.nom FROM (découverte d JOIN tableau t ON d.symbole =
    t.symbole )
JOIN grandeurs g ON d.symbole=g.symbole
WHERE numéro_atomique < 50 AND temp_fusion < 0
```

permet de récupérer les noms des personnes qui ont découvert les éléments dont le numéro atomique est < 50 et qui ne sont pas solides à une température nulle. Le préfixe `d` devant `nom` est nécessaire pour distinguer l'attribut `nom` de la table `découverte` (le nom de la personne qui a découvert l'élément) de l'attribut `nom` de la table `tableau` (le nom de l'élément).

- ⋆ Dans le cas où il faut un couple d'attributs pour constituer une clé primaire d'une table, la jonction peut se faire sur ces **deux** attributs :

```
SELECT * FROM table_A JOIN table_B ON table_A.attribut_1 = table_B.
    attribut_1 AND table_A.attribut_2 = table_B.attribut_2
```

II L'algèbre relationnelle

L'algèbre relationnelle est le cadre formel permettant d'exprimer les relations et les requêtes.

1. Opérations ensemblistes

Les commandes `UNION`, `INTERSECT` et `EXCEPT` permettent d'exécuter les opérations de réunion, d'intersection ou de différence sur les tables que l'on crée.

```
SELECT * FROM t1 UNION SELECT * FROM t2      # enregistrements présents dans t1 ou t2 (
    sans répétition)
SELECT * FROM t1 INTERSECT SELECT * FROM t2      # enregistrements présents dans t1 et
    t2
SELECT * FROM t1 EXCEPT SELECT * FROM t2         # enregistrements présents dans t1 mais
     pas dans t2
```

Pour pouvoir utiliser ces opération, les tables doivent avoir le même schéma relationnel (mêmes attributs)

Exemple(s) :

- ⋆ `SELECT * FROM tableau WHERE ligne = 2 UNION SELECT * FROM tableau WHERE symbole < 'D'` renvoie la sous-table de `tableau` des éléments qui sont sur la deuxième ligne ou dont le symbole commence par une lettre avant D.
- ⋆ `SELECT * FROM tableau WHERE ligne = 2 EXCEPT SELECT * FROM tableau WHERE colonne= 1` renvoie la sous-table des éléments de la ligne 2 sauf celui de la colonne 1.

Le complémentaire s'obtient en utilisant `NOT IN` : la requête `SELECT nom FROM tableau WHERE nom NOT IN (SELECT nom FROM tableau WHERE bloc = 'f')` renvoie le nom des éléments qui ne sont pas dans le bloc `f`.

2. Opérations de l'algèbre relationnelle

Projection

On note $\pi_{A_1,\dots,A_p}(R)$ la projection d'une relation R suivant les attributs $A_1, \dots, A_p$: cela revient à ne conserver que certain attributs de la table sans répétition. La requête SQL correspondante est

```
SELECT DISTINCT A1,...Ap FROM table
```

Sélection

On note $\sigma_E(R)$ la sélection d'une relation R suivant une condition logique E. La requête SQL correspondante est

```
SELECT * FROM table WHERE condition E
```

Jointure

On note $R_1 \bowtie_E R_2$ la jointure (symétrique) des relations R_1 et R_2 avec la condition E. La requête SQL correspondante est

```
SELECT * FROM table1 JOIN table2 ON cond E
```

Produit cartésien

On note $R_1 \times R_2$ le produit cartésien des relations R_1 et R_2 : on obtient une table qui contient les colonnes des deux tables (avec éventuelle répétitions, les colonne de la deuxième table à droite). La requête SQL correspondante est

```
SELECT * FROM table1, table2
```

Une jointure est en fait obtenue en réalisant un produit cartésien puis une sélection : $R_1 \bowtie_E R_2 = \sigma_E(R_1 \times R_2)$.

Division cartésienne

La division cartésienne de R_1 par $R_2 \subset R_1$ (la deuxiéme table est donc une sous-table de la première) est la relation $R = R_1 \div R_2$, c'est la relation contenant les attributs de R_1 qui ne sont pas dans R_2 : elle est caractérisée par $x \in R_1 \div R_2$ si et seulement si $\forall y \in R_2, (x, y) \in R_1$. Il n'y a pas de requête SQL simple permettant de reproduire cette opération.

table R_1

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2
a_2	b_2	c_3	d_3
a_3	b_3	c_1	d_1
a_3	b_3	c_2	d_2

tableR_2

C	D
c_1	d_1
c_2	d_2

table$R_1 \div R_2$

A	B
a_1	b_1
a_3	b_3