MODÈLE RELATIONNEL ET LANGAGE SQL

Did you really name your son `ROBERT'); DROP TABLE students;--`?

Exploits of a Mom, XKCD

À la fin de ce chapitre, je sais :

- ☞ interpréter et utiliser un modèle relationnel de base de données
- ☞ utiliser les opérateurs de projection et de sélection sur un modèle simple (select from, where)
- ☞ utiliser les clefs primaires et étrangères dans une requête simple
- ☞ opérer une jointure interne entre plusieurs tables (join on)
- ☞ utiliser les fonctions d'agrégation pour un calcul simple (min, max, sum, avg, count)
- ☞ filtrer des agrégations d'après un critère (having)
- ☞ utiliser des opérateurs ensemblistes (intersect, union, except)

A Requêtes SQL

SQL est le langage de manipulation et de création du modèle relationnel. Il implémente les opérations de l'algèbre relationnelle. De type déclaratif, ce langage décrit donc ce que l'on veut faire sur la base de données, pas comment on veut le faire car le SGBD s'en charge.

Les épreuves d'informatique commune proposent essentiellement de rédiger des requêtes SQL sur un base de données proposée. C'est un sous-ensemble de possibilité du langage qui permet déjà de se divertir un peu !

■ **Définition 1 — Requête SQL.** Une requête SQL consiste à extraire des données issues d'une base de données relationnelle en utilisant les opérateurs relationnels. Ces opérateurs sont :

- SELECT ... FROM
- WHERE

- GROUP BY
- HAVING
- ORDER BY

S'ajoutent à ces opérateurs, les opérations ensemblistes (union, intersection et différence) ainsi que l'opérateur de jointure (JOIN).

La structure générique d'une requête SQL est donnée sur le figure 1.

Code 1 – Structure de base d'une requête SQL : projection et sélection

```
1  -- Commentaire
2  SELECT nom              -- OPERATEUR   attribut
3  FROM ville              -- OPERATEUR  table
4  WHERE nom_pays = "France";  --OPERATEUR condition
```

Dans tout le chapitre, on considère qu'on manipule le modèle relationnel donné sur la figure 1 via le langage SQL.

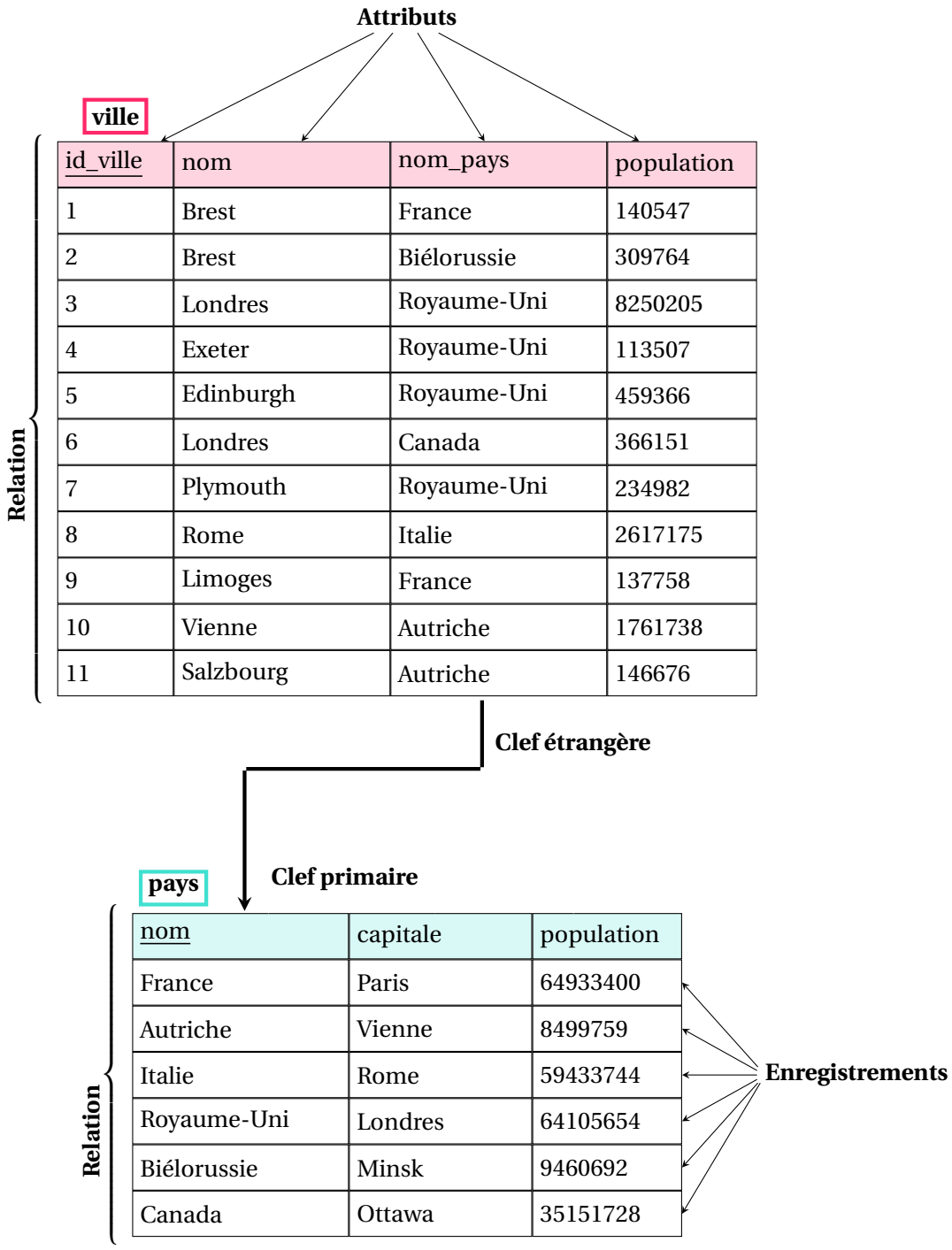

FIGURE 1 – Modèle relationnel utilisé pour les exemples de projection et de sélection.

B Projection : SELECT ...FROM

■ **Définition 2 — Projection.** Une projection est une opération qui consiste à sélectionner des colonnes (attributs) dans une table. En langage SQL, la projection est effectuée grâce aux mots clefs SELECT et FROM.

Code 2 – Exemples de projections

```
SELECT nom         -- la colonne nom
FROM ville;        --  de la table ville
-- SORTIE DU SGBD :
  -- Brest
  -- Brest
  -- Londres
  -- Exeter
  -- Edinburgh
  -- Londres
  -- Plymouth
  -- Rome
  -- Limoges
  -- Vienne
  -- Salzbourg

SELECT *         -- toutes les colonnes
FROM ville;      -- de la table ville
-- SORTIE DU SGBD :
  --  1 Brest France  140547
  --  2 Brest Bielorussie 309764
  --  3 Londres Royaume-Uni 8250205
  --  4 Exeter  Royaume-Uni 113507
  --  5 Edinburgh Royaume-Uni 459366
  --  6 Londres Canada  366151
  --  7 Plymouth  Royaume-Uni 234982
  --  8 Rome  Italie  2617175
  --  9 Limoges France  137758
  --  10  Vienne  Autriche  1761738
  --  11  Salzbourg Autriche  146676

SELECT nom, nom_pays     -- la colonne nom et la colonne nom_pays
FROM ville;          -- de la table ville
-- SORTIE DU SGBD :
  -- Brest  France
  -- Brest  Bielorussie
  -- Londres  Royaume-Uni
  -- Exeter Royaume-Uni
  -- Edinburgh  Royaume-Uni
  -- Londres  Canada
  -- Plymouth Royaume-Uni
  -- Rome Italie
  -- Limoges  France
  -- Vienne Autriche
  -- Salzbourg  Autriche
```

C Sélection : WHERE ...

■ **Définition 3 — Sélection.** Une sélection est une opération qui consiste à sélectionner des lignes (enregistrements) dans une table. En langage SQL, la sélection est effectuée grâce au mot clef `WHERE` suivi d'une condition.

Code 3 – Sélections

```
1  SELECT nom          -- la colonne nom
2  FROM ville          --  de la table ville
3  WHERE nom_pays = "France";
4  -- condition de selection : les lignes dont la colonne nom_pays vaut "
       France"
5  -- SORTIE DU SGBD :
6    -- Brest
7    -- Limoges
8
9  SELECT nom          -- la colonne nom
10 FROM ville          --  de la table ville
11 WHERE nom_pays = "France" OR nom_pays = "Royaume-Uni";
12 -- condition de selection : les lignes dont la colonne nom_pays vaut "
       France" ou du "Royaume Uni"
13 -- SORTIE DU SGBD :
14   -- Brest
15   -- Londres
16   -- Exeter
17   -- Edinburgh
18   -- Plymouth
19   -- Limoges
```

Une fois la projection et la sélection effectuée, on peut préciser la manière dont les résultats s'affichent : on peut ordonner les résultats selon un attribut en particulier, c'est la vocation des mots-clefs `ORDER BY`. L'ordre suivi par le SGBD sera l'ordre alphabétique ou numérique en fonction du type de données par rapport auquel on souhaite ordonner. Si on souhaite ordonner par un attribut de type texte, c'est l'ordre lexicographique qui est utilisé. Si on souhaite ordonner par un attribut du type entier, c'est l'ordre numérique qui sera choisi.

On peut également indiquer qu'on veut une projection sans redondance, c'est à dire ne garder qu'un seul exemplaire de chaque résultat (cf. code 4) grâce aux mots-clefs `SELECT DISTINCT`.

L'opérateur `LIMIT n` permet de limiter le nombre de réponses aux n premières. L'opérateur `OFFSET n` permet d'écarter les n premières réponses. En combinant les deux, on peut sélectionner n'importe quelle plage de la réponse.

Code 4 – Sélections - ordre des résultats - projection non redondante

```
SELECT nom         -- la colonne nom
FROM ville         --  de la table ville
WHERE nom_pays = "France" OR nom_pays = "Royaume-Uni"
ORDER BY nom_pays;
-- condition de selection : colonne nom_pays vaut "France" ou du "Royaume
    Uni"
-- ordonner par nom_pays
-- SORTIE DU SGBD :
  -- Brest
  -- Londres
  -- Exeter
  -- Edinburgh
  -- Plymouth
  -- Limoges

SELECT DISTINCT nom_pays    -- la colonne nom_pays
FROM ville                  --  de la table ville
ORDER BY nom_pays;
-- projection : les pays des villes sans redondance DISTINCT
-- ordonner par nom_pays
-- SORTIE DU SGBD :
  -- Autriche
  -- Bielorussie
  -- Canada
  -- France
  -- Italie
  -- Royaume-Uni

SELECT nom,  population   -- la colonne population
FROM ville                --  de la table ville
ORDER BY nom_pays         -- regrouper par pays
LIMIT 3;                  -- les trois premiers resultats
-- SORTIE DU SGBD :
  -- Vienne 1761738
  -- Salzbourg  146676
  -- Brest  309764

SELECT nom, population
FROM pays
ORDER BY population
LIMIT 3          -- les trois premiers resultats
OFFSET 2;     -- on ecarte les 5 premieres
-- SORTIE DU SGBD :
  -- Canada 35151728
  -- Italie 59433744
  -- Royaume-Uni  64105654
```

R Il est possible qu'il existe une ambiguïté dans le nom des attributs. Par exemple, la table *ville* possède un champ nom, tout comme la table *pays*. Parfois, cette ambiguïté n'a pas de conséquence, comme dans le code 10 : on a deux commandes `SELECT nom` qui désignent deux attributs différents, mais cela reste intelligible pour le SGBD, car ces commandes font parties de deux projections différentes, les `FROM` sont différents.

Cependant, il arrive que le SGBD ne puisse pas lever l'ambiguïté des noms utilisés. Dans ce cas, on peut soit :

- utiliser le préfixe du nom de la table suivi d'un point pour désigner un attribut. Par exemple, pour ne pas confondre la population d'une ville et la population d'un pays, on peut écrire les deux expressions `ville.population` et `pays.population`.
- renommer les tables, les attributs ou les champs calculés grâce à l'opérateur de renommage `AS` comme dans le code 5.

D Jointure : `JOIN …ON …`

Il est possible d'effectuer une requête SQL sur plusieurs tables simultanément, soit en utilisant le produit cartésien de tables soit en effectuant un jointure interne. Celle-ci est la syntaxe à privilégier pour cette opération.

■ **Définition 4 — Jointure.** En langage SQL, une jointure est un moyen d'utiliser plusieurs tables dans une même requête en créant une liaison entre les tables. Pour associer deux tables dans une jointure, il est nécessaire d'utiliser les mots-clefs `JOIN` et `ON`. Après `JOIN` on précise quelles tables sont à joindre, après `ON` on explicite la condition qui crée la liaison entre les tables.

■ **Exemple 1 — Exemples de jointures.** Le code 5 présente deux jointures de type différent.

La première jointure est une jointure entre deux tables différentes. La condition de jointure est que les lignes des deux tables qui sont à joindre doivent porter les mêmes noms de pays. La figure 2 détaille les étapes de cette requête.

La seconde est une autojointure, c'est à dire qu'on relie la table à elle même pour faire une requête. Dans cet exemple, on joint deux à deux les lignes des villes qui font parties d'un même pays, puis on sélectionne les lignes dont la première ville possède une population plus grande que la seconde.

Code 5 – Exemples de jointures

```
SELECT ville.nom,  pays.population
FROM ville
JOIN pays
ON ville.nom_pays = pays.nom
WHERE pays.population < 60000000;
-- Jointure de ville et pays d'apres les noms des pays
-- Condition : les pays comportent moins de 60 millions d'habitants
-- Sortie du SGBD
  -- Brest  9460692
  -- Londres  35151728
  -- Rome 59433744
  -- Vienne 8499759
  -- Salzbourg  8499759


SELECT v.nom, w.nom, v.population-w.population
FROM ville  as v
JOIN ville as w
ON v.nom_pays = w.nom_pays
WHERE v.population > w.population;
-- Projection avec renommage v
-- Autojointure avec renommage w
-- La condition de jointure est que le nom du pays des deux villes doit
    etre le meme.
-- La selection ne retient que les couples de villes tels que  la
    premiere ville est plus peuplee
-- On calcule la difference de population entre les deux villes
-- SORTIE DU SGBD :
  -- Exeter Edinburgh 345859
  -- Exeter Londres 8136698
  -- Exeter Plymouth  121475
  -- Edinburgh  Londres 7790839
  -- Plymouth Edinburgh 224384
  -- Plymouth Londres 8015223
  -- Limoges  Brest 2789
  -- Salzbourg  Vienne  1615062
```

ă

Étape 1 : jointure des tables ville et pays d'après les noms des pays.

id_ville	nom	nom_pays	population	nom	capitale	population
1	Brest	France	140547	France	Paris	64933400
2	Brest	Biélorussie	309764	Biélorussie	Minsk	9460692
3	Londres	Royaume-Uni	8250205	Royaume-Uni	Londres	64105654
4	Exeter	Royaume-Uni	113507	Royaume-Uni	Londres	64105654
5	Edinburgh	Royaume-Uni	459366	Royaume-Uni	Londres	64105654
6	Londres	Canada	366151	Canada	Ottawa	35151728
7	Plymouth	Royaume-Uni	234982	Royaume-Uni	Londres	64105654
8	Rome	Italie	2617175	Italie	Rome	59433744
9	Limoges	France	137758	France	Paris	64933400
10	Vienne	Autriche	1761738	Autriche	Vienne	8499759
11	Salzbourg	Autriche	146676	Autriche	Vienne	8499759

Étape 2 : sélection des pays de moins de 60 millions d'habitants.

id_ville	nom	nom_pays	population	nom	capitale	population
2	Brest	Biélorussie	309764	Biélorussie	Minsk	9460692
6	Londres	Canada	366151	Canada	Ottawa	3515172
8	Rome	Italie	2617175	Italie	Rome	59433744
10	Vienne	Autriche	1761738	Autriche	Vienne	8499759
11	Salzbourg	Autriche	146676	Autriche	Vienne	8499759

Étape 3 : projection du nom de la ville et de la population du pays.

nom	population
Brest	9460692
Londres	3515172
Rome	59433744
Vienne	8499759
Salzbourg	8499759

FIGURE 2 – Représentation des étapes de la requête `SELECT ville.nom, pays.population FROM ville JOIN pays ON ville.nom_pays = pays.nom WHERE pays.population < 60000000;` du code 5 qui opère une jointure entre les tables ville et pays. Étape 1 : les lignes des deux tables sont mises côtes à côtes d'après le critère de jointure. Étape 2 : seules les lignes quii respecte la condition de sélection sont sont gardées. Étape 3 : la projection sélectionne les colonnes résultats. ă

E Agréger, grouper et filtrer des résultats

a Fonctions d'agrégation : `SUM, COUNT, AVG, MAX, MIN`

■ **Définition 5 — Fonction d'agrégation.** Une fonction d'agrégation effectue des opérations statistiques sur un ensemble denregistrements. Elle sapplique à plusieurs lignes en même temps et permet :

- `COUNT` : de compter les éléments projetés et sélectionnés.
- `MIN` et `MAX` : de trouver le minimum ou le maximum des éléments,
- `SUM` : de calculer la somme des éléments,
- `AVG` : de calculer la moyenne des éléments.

R On peut utiliser les fonctions d'agrégation avec l'opérateur de projection `SELECT` ou l'opérateur `HAVING`. Par contre, **on n'utilise jamais** de fonctions d'agrégation avec l'opérateur de sélection `WHERE`. La raison est que pour pouvoir agréger des lignes, il faut d'abord sélectionner ces lignes des tables.

STOP Le paragraphe précédent est important pour l'épreuve d'informatique!

Code 6 – Exemples d'utilisation des fonctions d'agrégation

```
1  SELECT COUNT(nom_pays)
2  FROM ville
3  WHERE population >  1000000;
4  -- On compte les pays qui ont des villes de plus de  5 millions d'
       habitants.
5  -- SORTIE DU SGBD :
6    -- 3
7
8  SELECT MAX(population), nom
9  FROM ville
10 WHERE nom_pays = "Royaume-Uni";
11 -- On determine le nom de la ville du Royaume-Uni qui possede le plus
       grand nombre d'habitants.
12 -- SORTIE DU SGBD :
13   -- 8250205  Londres
14
15 SELECT MIN(nom)
16 FROM pays;
17 -- On determine le nom du pays le premier dans l'ordre lexicographique.
18 -- SORTIE DU SGBD :
19   -- Autriche
20
21 SELECT AVG(population), MIN(population), MAX(population)
22 FROM ville;
23 -- On determine la moyenne, le minimum et la maximum de la population des
        villes.
```

```
24  -- SORTIE DU SGBD :
25    -- 1321624.45454545 113507  8250205
```

b Regrouper : GROUP BY …

L'opérateur GROUP BY permet de créer des groupes dans la réponse. L'intérêt principal est que les fonctions d'agrégation peuvent calculer leurs statistiques non plus sur l'ensemble des réponses mais sur chacun de ces groupes.

Code 7 – Exemple d'utilisation de GROUP BY

```
1  SELECT nom_pays, SUM(population)
2  FROM ville
3  WHERE population > 200000
4  GROUP BY nom_pays;
5  -- On fait la somme des populations des villes de plus de 200000
       habitants par pays.
6  -- SORTIE DU SGBD :
7    -- Autriche 1761738
8    -- Bielorussie  309764
9    -- anada  366151
10   -- Italie 2617175
11   -- Royaume-Uni  8944553
```

c Filtrer des résultats regroupés : GROUP BY …HAVING…

SQL permet de filtrer les groupes crées par GROUP BY. En effet, l'opérateur WHERE a déjà sélectionné les lignes, les groupes ont été créés à partir de ces lignes. Si on souhaite filtrer les groupes, il faut donc un autre opérateur. C'est le rôle de HAVING.

Code 8 – Exemple d'utilisation de GROUP BY . . . HAVING

```
1  SELECT nom_pays, SUM(population)
2  FROM ville
3  WHERE population > 200000
4  GROUP BY nom_pays
5  HAVING COUNT(nom) > 2;
6  -- On fait la somme des populations des villes de plus de 200000
       habitants par pays
7  -- si seulement le pays en possede au moins 3.
8  -- SORTIE DU SGBD :
9    -- Royaume-Uni  8944553
```

F Opérations ensemblistes

SQL définit également des opérateurs ensemblistes : un opérateur d'union, d'intersection et de différence. Ils sont principalement utiles lorsqu'on dispose de plusieurs tables cohérentes,

voire qui représente la même entité, et qu'on souhaite extraire des informations de ces deux tables simultanément. Il est nécessaire que les projections génèrent le même nombre de colonnes pour pouvoir mener une opération ensembliste. Les opérateurs sont `UNION`, `INTER` et `EXCEPT`.

Code 9 – Exemple d'opération ensembliste

```
SELECT nom, population
FROM ville
WHERE nom = "Londres"
UNION
SELECT nom, population
FROM ville
WHERE population < 200000;
-- SORTIE DU SGBD :
  -- Brest  140547
  -- Exeter 113507
  -- Limoges  137758
  -- Londres  366151
  -- Londres  8250205
  -- Salzbourg  146676
```

G Requêtes imbriquées --→ HORS PROGRAMME

■ **Définition 6 — Sous-requête ou requête imbriquée.** Une requête imbriquée est une requête incluse dans une requête. Son résultat devient le champ de recherche d'une autre requête.

Vocabulary 1 — Nested request ⇜⇝ Requête imbriquée

Comme le montre le code 10, il est possible d'utiliser les opérateurs de comparaison `=`, `<`, `<=`, `>=`, `<>` pour créer une condition qui dépend de la requête imbriquée, si celle-ci ne renvoie qu'une seule valeur. Mais ce n'est pas toujours le cas. Dans le cas contraire, la sous-requête résulte en une liste de valeur, alors il est nécessaire d'utiliser les mots-clefs `IN` ou `NOT IN` pour tester l'appartenance du critère à la liste de valeurs retournée par la requête imbriquée.

Code 10 – Requêtes imbriquées

```
SELECT nom
FROM ville
WHERE ville.pays = ( SELECT nom
                     FROM pays
                     WHERE capitale = "Paris");
-- Requete imbriquee : le noms du pays dont la capitale est Paris
-- Sortie du SGBD
-- Brest
-- Limoges

```

```sql
SELECT nom
FROM ville
WHERE pays IN  ( SELECT nom
                 FROM pays
                 WHERE population < 60000000);
-- Requete imbriquee : la liste des noms des pays qui comportent moins de
     60 millions d'habitants
-- Sortie du SGBD
-- Brest
-- Londres
-- Rome
-- Vienne
-- Salzbourg
```

H De belles requêtes SQL

Comme vous l'aurez observé tout au long de ce chapitre, on n'écrit pas n'importe comment les requêtes SQL. Les codes qui vont sont présentés respectent certaines conventions.

Méthode 1 — Écriture de requêtes SQL et conventions Afin de garantir une bonne lisibilité des requêtes SQL, il est nécessaire de bien les écrire. Même s'il ne s'agit de convention, il est important :

- d'écrire en majuscules pour les opérateurs SQL,
- d'écrire en minuscules les relations (tables) et les attributs (champs),
- de revenir à la ligne après chaque opération : projection, sélection, regroupement, agrégation, ordonnancement.

Le paragraphe précédent est important pour l'épreuve d'informatique!

Opérateurs	Action
`SELECT ... FROM ...`	Projection des colonnes d'une table
`SELECT DISTINCT ... FROM ...`	Idem mais sans redondance, sans doublons
`WHERE ...`	Condition de sélection des lignes
`GROUP BY ...`	Créer des regroupements des résultats
`HAVING ...`	Filtrer les regroupements de résultats
`ORDER BY ... ASC/DESC`	Ordonner les résultats
`LIMIT n`	Limiter le nombre de résultats aux n premiers
`OFFSET n`	Écarter les n premiers résultats
`UNION, INTERSECT, EXCEPT`	Opérations ensemblistes

TABLE 1 – Synthèse des opérateurs SQL au programme