Aide Mémoire – Bases de données

1. Algèbre relationnelle et SQL

- **Projection** sur $(A_1,\dots,A_n)$ de R : $\Pi_{(A_1,\dots,A_n)}(R)$.

```
SELECT A1, ..., An FROM R;
```

- **Selection** selon le prédicat p de R : $\sigma_p(R)$.

```
SELECT * FROM R WHERE p;
```

- **Opérations ensemblistes** (même schéma) : $R_1 \cup R_2$, $R_1 \cap R_2$, $R_1 - R_2$.

```
SELECT ... INTERSECT SELECT ...;
SELECT ... UNION     SELECT ...;
SELECT ... UNION ALL SELECT ...; -- Avec les doublons
SELECT ... EXCEPT    SELECT ...; -- MINUS en MySQL
```

- **Jointure** de R_1 et R_2 selon $R_1.X = R_2.Y$: $R_1 \underset{R_1.X=R_2.Y}{\bowtie} R_2$.

```
SELECT * FROM R1 JOIN R2 ON R1.X = R2.Y; -- A préférer
SELECT * FROM R1, R2 WHERE R1.X = R2.Y;
```

- **Renommage** de X en A dans R : $\rho_{X\to A}(R)$.

```
SELECT R.X AS A, ... FROM R;
```

- **Produit cartésien** de R_1 et R_2 : $R_1 \times R_2$.

```
SELECT * FROM R1, R2;
```

- **Division cartésienne** de R_1 et R_2 (schéma de $R_2 \subset$ schéma de R_1) : $R_1 \div R_2$ (tuples qui, concaténés à ceux de R_2 apparaissent tous dans R_1.)

2. Interrogation de la base de données

- **Requêtes de base** :

```
SELECT ... FROM ... WHERE ...;
SELECT DISTINCT ... FROM ... WHERE ...;
SELECT ... JOIN ... ON ... WHERE ...;
SELECT ... FROM ... WHERE ... ORDER BY ... ;
SELECT ... FROM ... WHERE ... ORDER BY ... DESC;
SELECT ... FROM ... WHERE ... LIMIT ... OFFSET ...;
```

- **Renommages** :

```
SELECT A1 AS nom_1, ..., An AS nom_n FROM table1 nomTable1 ...;
```

- **Conditions** : `=` ; `!=` ou `<>` ; `<` ; `>` ; `<=` ; `>=` ; `IS NULL` ; `IS NOT NULL` ; `AND` ; `OR` ; `NOT` ; `BETWEEN ... AND ...` ; `LIKE '...'` (`_` remplace un caractère, `%` remplace un nombre quelconque de caractères.) ; `IN` ; `NOT IN` ; (comparateur) `ANY` ; (comparateur) `ALL` .

- **Fonctions d'agrégat** : `SUM`, `MIN`, `MAX`, `AVG`, `COUNT`.

```
SELECT fonction(attribut(s)) FROM ...;
```

(ne renvoie qu'un résultat.)

- **Existence**

```
SELECT ... FROM ... WHERE EXISTS(SELECT ...);
SELECT ... FROM ... WHERE NOT EXISTS(SELECT ...);
```

- **Regroupement**

```
SELECT ... FROM ... WHERE ... GROUP BY ...;
SELECT ... FROM ... WHERE ... GROUP BY ... HAVING ...;
```

Le contenu du `SELECT` doit être des attributs du `GROUP BY` ou des attributs identiques pour chaque élément du groupe ou des fonctions d'agrégat qui porteront sur chaque groupe.

Le contenu du `HAVING` porte sur des attributs n'apparaissant pas dans le `GROUP BY`.

Renvoie un résultat par groupe.

- **Agrégat de double niveau**

```
SELECT fonction2(attribut2)
FROM (SELECT fonction1(attribut1) AS attribut2
      FROM ... GROUP BY ...);
```

3. Création et modification des tables (en SQLite)

- **Création de table**

```
CREATE TABLE nomTable (
    nomAtt1 typeAtt1 [PRIMARY KEY] [REFERENCES nomTable[(att)]],
    ...
    nomAttn typeAttn [REFERENCES nomTable[(att)]],
    [PRIMARY KEY (Att1, ... ,Attp),]
    [FOREIGN KEY (Att1, ... ,Attp) REFERENCES nomTable[(att1,...,attp)],]
    [CHECK (condition),]
);
```

Les arguments entre crochets sont facultatifs. Si plus d'un attribut dans la clé primaire, la déclarer à part.
`typeAtt` parmi `INTEGER`, `REAL`, `TEXT`.

- **Suppression d'une table**

```
DROP TABLE nomTable; -- Bien réfléchir avant de l'utiliser !!
```

- **Création d'un tuple**

```
INSERT INTO nomTable VALUES (val_1, ... , val_n);
```

- **Suppression d'un tuple**

```
DELETE FROM nomTable WHERE condition;
```

- **Modification d'un tuple**

```
UPDATE nomTable SET attr_1 = val_1, ..., attr_n = val_n
WHERE condition;
```