Requêtes SQL et mise à jour

A l'aide du langage SQL, nous avons vu précédemment :

- Comment créer des tables cohérentes, qui tiennent compte des contraintes d'intégrité.
- Les remplir.

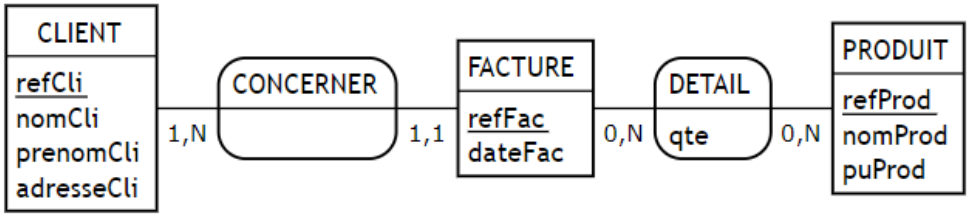
Dans ce document , nous allons aborder deux autres utilisations courantes d'un Système de Gestion de Base de Données (SGBD):

- La sélection de données.
- La mise à jour de données.

Ce document fait des aller-retours réguliers avec la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

Exemple 1 :

Nous allons reprendre la base de données précédemment construite à partir du MCD(Modèle Conceptuel de Données) et MLD(Modèle Logique de Données) ci-dessous :



CLIENT (refCli, nomCli, prenomCli, adresseCli)
FACTURE (refFac, dateFac, *refCli*)
DETAIL (*refFac*, *refProd*, qte)
PRODUIT (refProd, nomProd, puProd)

Les scripts de création et de peuplement des tables nous ont permis d'obtenir les tables ci-desous :

1 *CLIENT*

refCli	nomCli	prenomCli	adresseCli
1	BANAZIAI	Jules	Grenoble
2	DONGEPLI	Armelle	Limoges
3	BOBAINO	Julie	Nimes
4	BOUTIDICHT	Maxime	Grenoble

2 *PRODUIT*

refProd	nomProd	puProd
1	Sucre	1.20
2	Cereales	1.30
3	Biscottes	1.15
4	Poudre petit dejeuner	2.30
5	Cafe	2.50
6	The	3.10

3

refFac	refProd	qte
1	2	8
1	3	2
2	6	7
3	1	2
3	3	2
3	5	9

DETAIL

4

refFac	dateFac	refCli
1	2019-06-14	2
2	2019-12-26	2
3	2019-03-14	1

FACTURE

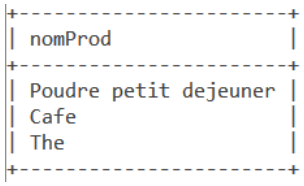
1. Sélection

- Une première utilisation courante d'une base est d'en extraire des données vérifiant certains critères. (les clients habitant une certaine ville, les produits dont le prix est supérieur à une certaine valeur, le contenu d'une facture,etc...).
- Le programmeur de base de données consiste alors à traduire les questions naturelles que l'on se pose, en langage SQL pour que le SGBD puisse y répondre.

Requête sur une table (SELECT...FROM...WHERE...)

Exemple 2 :

- **Question :** Quels sont les noms des produits dont le prix unitaire est supérieur à 2 euros ?
- **Ordre SQL:** `SELECT nomProd FROM PRODUIT WHERE puProd > 2;`
- **Réponse du système:**



```
+------------------------+
| nomProd                |
+------------------------+
| Poudre petit dejeuner |
| Cafe                   |
| The                    |
+------------------------+
```

Remarques :

- Dans cette requête, `FROM PRODUIT` indique la table concernée par la requête.
- `WHERE...` indique que l'on ne sélectionne que les lignes de la table `PRODUIT` pour lesquelles la valeur de l'attribut `nomProd` est supérieure à 2.
- `SELECT nomProd` indique que l'on veut uniquement le nom des produits.
- **Important :**
  - Le résultat de la requête `SELECT` est une table. Dans cet exemple, on obtient une table avec une unique colonne `nomProd` .

Clause `WHERE`

- Elle permet de restreindre les lignes de la table.
- L'expression qui se trouve dans la partie `WHERE` doit être une expression booléenne. Elle peut être construite, entre autre, à l'aide de:
  - Des opérateurs de comparaison: `<` , `>` , `=` , `<=` , `>=` et `<>` ou `!=` pour "différent de" suivant les systèmes.
  - Des opérateurs arithmétiques: `+` , `-` , `*` , `/` , etc...
  - Des opérateurs logiques : `AND` , `OR` , `NOT` .
  - D'autres opérateurs spéciaux...

Exemple 3 :

- **Question :** Quels sont les noms des produits dont le prix unitaire est compris entre 1,25 et 2,50 euros inclus ?
- **Ordre SQL:** `SELECT nomProd FROM PRODUIT WHERE puProd>= 1.25 AND puProd<= 2.5;`
- **Réponse du système:**

```
+-----------------------+
| nomProd               |
+-----------------------+
| Cereales              |
| Poudre petit dejeuner |
| Cafe                  |
+-----------------------+
```

Clause `SELECT`

- Elle permet de restreindre les colonnes de la table.
  - Les colonnes que l'on veut obtenir sont listées après `SELECT` , séparées par une virgule.
  - ##### Exemple 4:
    - **Question :** Quels sont les noms et les prix unitaires des produits dont le prix unitaire est supérieur à 2 euros ?
    - **Ordre SQL:** `SELECT nomProd,puProd FROM PRODUIT WHERE puProd >=2;`
    - **Réponse du système:**

```
+-----------------------+--------+
| nomProd               | puProd |
+-----------------------+--------+
| Poudre petit dejeuner |   2.30 |
| Cafe                  |   2.50 |
| The                   |   3.10 |
+-----------------------+--------+
```

  - Si l'on veut récupérer toutes les colonnes, on peut utiliser le symbole `*` après `SELECT` .
  - ##### Exemple 5:
    - **Question :** Quels sont les produits dont le prix unitaire est supérieur à 2 euros ?
    - **Ordre SQL:** `SELECT * FROM PRODUIT WHERE puProd >=2;`
    - **Réponse du système:**

```
+---------+-----------------------+--------+
| refProd | nomProd               | puProd |
+---------+-----------------------+--------+
|       4 | Poudre petit dejeuner |   2.30 |
|       5 | Cafe                  |   2.50 |
|       6 | The                   |   3.10 |
+---------+-----------------------+--------+
```

- Ele permet aussi d'appeler des fonctions pour effectuer par exemple des calculs(somme, moyenne, comptage, minimum, maximum...) et de renvoyer le résultat dans une table avec une ligne et une colonne.
  - ##### Exemple 6:
    - **Question :** Quel est le prix unitaire moyen des produits proposés ?
    - **Ordre SQL:** `SELECT AVG(puProd) FROM PRODUIT ;`
    - **Réponse du système:**

```
+-------------+
| avg(puProd) |
+-------------+
|    1.925000 |
+-------------+
```

  - ##### Exemple 6bis:
    - Il est possible de renommer la ou les colonnes de la table obtenue avec `AS`
    - **Ordre SQL:** `SELECT AVG(puProd) AS prix_moyen FROM PRODUIT ;`
    - **Réponse du système:**

```
+------------+
| prix_moyen |
+------------+
|   1.925000 |
+------------+
```

Exercice 1 : Sélection des données

Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

Tri, gestion des doublons (ORDER BY..., DISTINCT...)

- Les résultats des requêtes ne sont pas toujours affichés dans le même ordre. En fonction de certains paramètres, le SGBD peut choisir entre différentes façons de calculer les requêtes. L'affichage peut varier entre plusieurs éxécutions d'une même requête.Pour gérer cela , on peut demander un classement des résultats à l'aide de `ORDER BY` .
  - ##### Exemple 7:
    - **Demande :** Les noms et les prix des produits , classés par ordre alphabétique.
    - **Ordre SQL:** `SELECT nomProd, puProd FROM PRODUIT ORDER BY nomProd ASC;`
    - **Réponse du système:**

```
+-----------------------+--------+
| nomProd               | puProd |
+-----------------------+--------+
| Biscottes             |   1.15 |
| Cafe                  |   2.50 |
| Cereales              |   1.30 |
| Poudre petit dejeuner |   2.30 |
| Sucre                 |   1.20 |
| The                   |   3.10 |
+-----------------------+--------+
```

  - ##### Exemple 8:
    - **Demande :** Les noms et les prix des produits, classés par prix décroissants.
    - **Ordre SQL:** `SELECT nomProd, puProd FROM PRODUIT ORDER BY puProd DESC;`
    - **Réponse du système:**

```
+-----------------------+--------+
| nomProd               | puProd |
+-----------------------+--------+
| The                   |   3.10 |
| Cafe                  |   2.50 |
| Poudre petit dejeuner |   2.30 |
| Cereales              |   1.30 |
| Sucre                 |   1.20 |
| Biscottes             |   1.15 |
+-----------------------+--------+
```

- Dans les résultats d'une requête peuvent apparaître des lignes en plusieurs exemplaires (âr exemple, les villes des clients).Pour retirer les doublons, on ajoute le mot-clé `DISTINCT` à la clause `SELECT` .
  - ##### Exemple 9:
    - **Demande :** Les villes des clients, sans doublons, classées par ordre alphabétique.
    - **Ordre SQL:** `SELECT DISTINCT adresseCli FROM client ORDER BY adresseCli ASC;`
    - **Réponse du système:**

```
+------------+
| adresseCli |
+------------+
| Grenoble   |
| Limoges    |
| Nimes      |
+------------+
```

Exercice 2 : Tris et doublons

Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

Jointure (JOIN)

- Les requêtes que nous avons vues jusqu'à présent concernaient à chaque fois une et une seule des quatre tables.
- Ainsi lorsque l'on veut les clients qui ont acheté des produits avec la commande `select distinct refcli from facture;` , on obtient la référence des clients mais pas leur nom, ce qui semble pourtant plus naturel.
- L'opération de jointure de deux tables va permettre de remédier à ce problème.
- ##### Exemple 10:
  - **Demande :** Les clients qui ont acheté des produits.
  - **Ordre SQL:** `SELECT * FROM CLIENT JOIN FACTURE ON CLIENT.refCli=FACTURE.refCli ;`
  - **Réponse du système:**

refCli	nomCli	prenomCli	adresseCli	refFac	dateFac	refCli
1	BANAZIAI	Jules	Grenoble	3	2019-03-14	1
2	DONGEPLI	Armelle	Limoges	1	2019-06-14	2
2	DONGEPLI	Armelle	Limoges	2	2019-12-26	2

Remarques:

- Cette requête crée la jointure des deux tables CLIENT et FACTURE affichée ci-dessus.
- Chaque ligne est le résultat de la fusion de deux lignes ayant la même référence client.
- Le choix de ces lignes est donné par la condition de jointure indiquée par le mot-clé `ON` .
- La notation avec le point permet de différencier la provenance de deux attributs ayant le même nom. Plus généralement , on peut utilisee cette syntaxe pour éviter les ambiguités.
- La jointure peut bien sûr être combinée avec les clauses `SELECT` et `WHERE` .
- On peut effectuer des jointures sur plus de deux tables :
  - ##### Exemple 11:
    - **Demande :** Les dates d'achat des produits.
    - **Ordre SQL:**
      ```
      SELECT * FROM FACTURE
      JOIN DETAIL ON FACTURE.refFac=DETAIL.refFac
      JOIN PRODUIT ON DETAIL.refProd=PRODUIT.refProd;
      ```
    - **Réponse du système:**

refFac	dateFac	refCli	refFac	refProd	qte	refProd	nomProd	puProd
1	2019-06-14	2	1	2	8	2	Cereales	1.30
1	2019-06-14	2	1	3	2	3	Biscottes	1.15
2	2019-12-26	2	2	6	7	6	The	3.10
3	2019-03-14	1	3	1	2	1	Sucre	1.20
3	2019-03-14	1	3	3	2	3	Biscottes	1.15
3	2019-03-14	1	3	5	9	5	Cafe	2.50

  - On joint ici les tables `DETAIL` et `FACTURE` à l'aide de l'attribut `refFac` , puis la table obtenue avec la table `PRODUIT` à l'aide de l'attribut `refProd` .

Exercice 3 : Jointures

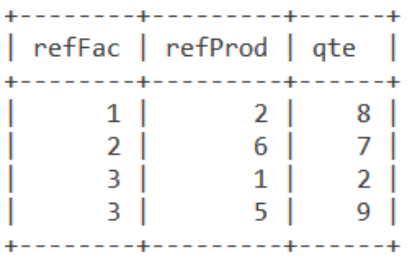
Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

2. Mise à jour

- Les données dans un Système de Gestion de Base de Données n'ont pas vocation à être figées et peuvent être modifiées dans le temps par les utilisateurs autorisés.
- Nous allons aborder deux types de modification :
  - La suppression d'en ensemble de lignes
  - La mise à jour de certains attributs dans un ensemble de lignes.

Suppression de lignes (DELETE FROM...WHERE...)

- ##### Exemple 12:
  - **Demande :** Supprimer les biscottes(produit n°3) de toutes les factures.
  - **Ordre SQL:** `DELETE FROM DETAIL WHERE refProd=3;`
  - **Vérification:** `SELECT * FROM DETAIL;`
  - **Réponse du système:**



refFac	refProd	qte
1	2	8
2	6	7
3	1	2
3	5	9

Remarques :

- Attention : Si aucune clause `WHERE` n'est spécifiée, on efface **toutes** les lignes de la table.
  - Ainsi, l'ordre `DELETE FROM DETAIL;` efface le contenu de toutes les factures(mais pas la table).

- Les contraintes d'intégrité sont vérifiées à chaque mise à jour
  - ##### Exemple 13:
    - **Demande :** Supprimer le thé de la liste des produits disponibles.
    - **Ordre SQL:** `DELETE FROM PRODUIT WHERE nomProd='The';`
    - **Réponse du système:** `Cannot delete or update a parent row: a foreign key constraint fails (`facture`.`detail`, CONSTRAINT `detail_ibfk_1` FOREIGN KEY (`refProd`) REFERENCES `produit` (`refProd`))`
  - Le système nous indique ici que la suppression du thé de la liste des produits viole la contrainte de clé étrangère dans la table `DETAIL` , puisque le thé est présent dans au moins une des factures.La suppresion est donc annulée.
  - Il faut d'abord supprimer en premier les lignes ou le l'attribut `refProd` est déclaré comme clé étrangère(c'est le même principe que pour la suppresion de tables).
- Un ordre de modification est soit entièrement exécuté, soit entièrement annulé.
  - ##### Exemple 14:
    - **Demande :** Supprimer les clients qui habitent Grenoble.
    - **Ordre SQL:** `DELETE FROM CLIENT WHERE adresseCli='Grenoble';`
    - **Réponse du système:** `Cannot delete or update a parent row: a foreign key constraint fails (`facture`.`facture`, CONSTRAINT `facture_ibfk_1` FOREIGN KEY (`refCli`) REFERENCES `client` (`refCli`))`
  - Le client n°1 a acheté des produits, sa référence est donc présente comme clé étrangère dans la table `FACTURE` .Bien que la référence de l'autre client habitant Grenoble n'apparaisse pas dans la table `FACTURE` , ses données ne seront pas effacées, car l'ordre sera entièrement annulé.

Exercice 4: Suppressions de lignes

Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

Mise à jour d'attributs (UPDATE...SET...)

- ##### Exemple 13:
  - **Demande :** Le prix du sucre a augmenté de 10% et s'appelle désormais "Sucre bio". Mettre à jour la base de données.
  - **Ordre SQL:** `UPDATE PRODUIT SET nomProd='Sucre bio',puProd=1.32 where nomProd='Sucre';`
  - **Vérification:** `SELECT * FROM PRODUIT;`
  - **Réponse du système:**

```
+---------+-----------+--------+
| refProd | nomProd   | puProd |
+---------+-----------+--------+
|       1 | Sucre bio |   1.32 |
|       2 | Cereales  |   1.30 |
|       3 | Biscottes |   1.15 |
|       5 | Cafe      |   2.50 |
|       6 | The       |   3.10 |
+---------+-----------+--------+
```

- #### Remarques :
  - On indique la table concernée après la clause `UPDATE` .
  - L'affectation des nouvelles valeurs se fait après `SET` les attributs à modifier sont séparés par des virgules.
  - La clause `WHERE` permet de spécifier l'article concerné à l'aide d'un booléen.
  - Syntaxe:
    - Le langage SQL comporte des similarités avec certains langages de programmation :
      - On peut par exemple utiliser l'affectation `puProd=puProd*1.1` pour augmenter le prix de 10%
      - De façon séquentielle :
        - `nomProd='Sucre'` signifie que l'on évalue le contenu courant de la variable `nomProd` .
        - `nomProd='Sucre bio'` signifie que l'on affecte une nouvelle valeur à cette variable.

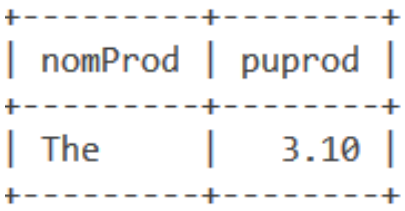
Exercice 5: Mise à jour d'attributs.

Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

3. Requêtes imbriquées

Le résultat d'une clause `SELECT` est une table. On peut donc effectuer des requêtes sur cette nouvelle table !

- ##### Exemple 14:
  - **Demande :** Le nom du produit dont le prix unitaire est le plus cher.
  - **Ordre SQL:** `SELECT nomProd,puProd from PRODUIT WHERE puProd=( SELECT MAX(puProd) from PRODUIT);`
  - **Réponse du système:**



nomProd	puprod
The	3.10

Remarques :

- Ici la sous-requête `SELECT MAX(puProd) from PRODUIT;` renvoie une table avec une valeur ce qui permet ensuite l'évaluation de `puProd=...`
- Les sous-requêtes peuvent aussi se trouver après la clause `FROM` .

Exercice 6 : Requêtes imbriquées

Il se trouve dans la feuille d'exercices (Exercices/requetes_sql_exos.pdf).

4. En savoir plus

- Les quelques notions présentées ici et dans les précédents chapitres sur les bases de données constituent une bonne initiation au langage SQL.
- Cependant, son utilisation avancée nécéssite un apprentissage approfondi, le lecteur intéréssé pourra consulter les liens suivant pour en savoir plus :
  - https://sql.sh/ (https://sql.sh/) : Cours et tutoriels en français.
  - http://webtic.free.fr/sql/exint/q1.htm (http://webtic.free.fr/sql/exint/q1.htm) : pour s'entraîner aux requêtes
  - https://www.w3schools.com/sql/ (https://www.w3schools.com/sql/) : apprende et s'entraîner, en français (traduction automatique pas toujours très claire)