TD n°11 : Ma première Base de Données

4 heures

Rédigé par Pascal Delahaye

18 mai 2018

Le but de ce TD est de découvrir les principales fonctions d'OpenOffice Base, le système de gestion (création et utilisation) de bases de données proposé dans la suite OpenOffice disponible gratuitement sur Internet.

Nous allons successivement :

1. Apprendre à créer une base et les tables de données
2. Apprendre à utiliser les fonctions :
   (a) d'opérations ensemblistes : union, intersection, soustraction et produit cartésien de deux tables
   (b) de sélection dans une table
   (c) de projection d'une table
3. Apprendre à relier des tables entre elles par des relations et effectuer des jointures entre tables
4. Apprendre à utiliser des fonctions d'agrégation

Parmi les objectifs de ce TD, s'ajoute également la familiarisation avec la formulation en langage SQL de requêtes d'abord simples puis plus complexes.

1 Création d'une base de données

Pour la création de la nouvelle base, effectuer les manipulations suivantes :

1. *Ouvrir OpenOffice Base*
2. *Sélectionner "Créer une base de données"*
3. *Sélectionner "Non, je ne souhaite pas que la base de données soit référencée" puis "terminer"*
4. *Enregistrer votre nouvelle base de données dans votre espace personnel sous le nom "fidelite"*

Voilà... Vous venez de créer votre première base de données... Vide pour l'instant...

Il reste alors :

1. à créer les tables de données
2. à extraire des informations relatives à ces tables (requêtes)

1.1 Création des tables

Pour créer une table, on commence par sélectionner l'icône "table" dans la partie gauche de l'écran.
Apparaît alors 2 façons possibles de créer une table :

1. soit en mode "ébauche"
2. soit à l'aide de l'assistant

Nous allons choisir le mode "ébauche" qui offre le plus de flexibilité.
Apparaît alors un tableau permettant de définir le schéma relationnel de la table.

Exercice : 1

Création de la structure des tables de données

Créer les 4 tables suivantes :
Pour indiquer qu'un attribut est une clé primaire (obligatoire sous OpenBase !), on clique avec le bouton droit de la souris à gauche du nom de l'attribut et on sélectionne "clé primaire".

1. La table "client1" de schéma relationnel :

   S1 = ((numclient, INTEGER (clé primaire - autovaleur = OUI)), (nomclient, CHAR (15 caractères - saisie requise = OUI)), (prenom, CHAR (15 caractères - saisie requise = OUI)), (age, INTEGER (saisie requise = OUI)), (sexe, CHAR (1 caractère)))

2. La table "client2" de schéma relationnel : S2 = S1 *(on pourra ici "copier" la table "client1")*

3. La table "produit" de schéma relationnel :

   S3 =((numproduit, INTEGER (clé primaire - autovaleur = OUI)), (nomproduit, CHAR (20 caractères - saisie requise = OUI)), (prix, NUMERIC (longueur = 5, décimale = 2 - saisie requise = OUI)), (stock, BOOL (saisie requise = OUI)))

4. La table "achat" de schéma relationnel :

   S4 = ((numachat, INTEGER (clé primaire - autovaleur = OUI)), (numclient, INTEGER (saisie requise = OUI)), (numproduit, INTEGER (saisie requise = OUI)), (date, DATE (saisie requise = OUI)), (promotion, BOOL (saisie requise = OUI)))

1.2 Insertion des données

La structure des tables étant construites, il nous faut maintenant entrer les données.
Pour cela, il suffit de cliquer sur l'icône de la table concernées et remplir le tableau qui apparaît à l'écran.

Exercice : 2

Insertion des données dans les tables

Remplisser les tables "client1", "client2" et "produit" avec les données suivantes...
Attention à bien respecter l'orthographe et en particulier les MAJUSCULES !

numclient	nomclient	prenom	age	sexe
0	Hachard	Samuel	34	H
1	Berthy	Anissa	23	F
2	Allard	Kevin	45	H
3	Courcelle	Aline	67	F
4	Kim	Sung-Ah	45	F
5	Ben chaid	Cherif	32	H
6	Keita	Salif	13	H
7	Dilan	Bob	78	H
8	Toto	Salif	13	H

"client1"

numclient	nomclient	prenom	age	sexe
0	Gomez	Antonio	47	H
1	Garland	Todd	14	H
2	James	Tania	30	F
3	Leer	Amanda	72	F
4	Pichon	Laure	24	F
5	Lin	Yu-Ting	27	F
6	Berthy	Anissa	23	F
7	Allard	Kevin	45	H

"client2"

numproduit	nomproduit	prix	stock
0	savon	2,30	☑
1	dentifrice	1,70	☑
2	shampoing	3,20	☐
3	rasoir	7,50	☑
4	mouchoir	1,60	☑
5	mousse	3,40	☐
6	parfum	10,30	☑

"produit"

2 Opérations ensemblistes usuelles

Nous allons ici effectuer nos premières requêtes.

Pour la création de requêtes, nous pouvons utiliser :
- Soit "créer une requête en mode ébauche..."
- Soit "Utiliser l'assistant de création de requêtes..."
- Soit "Créer une requête en mode SQL..."

Les deux premières options permettent à l'utilisateur de créer des requêtes, même si celui-ci ne connaît pas le langage SQL qui est le langage standard pour la création de requête. Les assistants de requêtes ne permettent pas d'effectuer des réunions, intersections et soustractions de tables. Ainsi, même si vous ne connaissez pas encore le langage SQL, nous sommes contraints dans cette partie de formuler nos requêtes directement en langage SQL (option 3). Ce sera donc l'occasion de commencer à vous familiariser avec ce langage.

Exercice : 3

Union - Intersection - Soustraction et Produit Cartésien

Nous souhaitons réunir les deux tables "client1" et "client2" en une même table.
Pour cette opération, OpenOffice Base nous contraint à utiliser le langage SQL :

Vous devez alors effectuer les manipulations suivantes :
- *Sélectionner la section "requête" dans la colonne à gauche de l'écran*
- *Sélectionner la tâche "Créer une requête en mode SQL..."*

1. UNION : *Taper dans la requête les instructions suivantes :*

SQL
```
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client1"
UNION
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client2"
```

   (a) *Tenter de comprendre la syntaxe du langage SQL.*
   (b) *Cliquer sur F5 ou sur l'icône "Exécuter la requête"*
   (c) *Enregistrer votre requête sous le nom de "unionclient"*

2. INTERSECTION : *Taper dans la requête les instructions suivantes :*

SQL
```
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client1"
INTERSECT
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client2"
```

   Enregistrer le résultat de la requête sous le nom "intersectionclient".

3. SOUSTRACTION : *Taper dans la requête les instructions suivantes :*

SQL
```
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client1"
EXCEPT
SELECT  "nomclient", "prenom", "age", "sexe"
FROM "client2"
```

Enregistrer le résultat de la requête sous le nom "soustractionclient" et vérifier le résultat obtenu.

4. PRODUIT CARTESIEN : *Taper dans la requête les instructions suivantes :*

SQL
```
SELECT  "client1"."nomclient", "produit"."nomproduit"
FROM "client1", "produit"
```

Enregistrer le résultat de la requête sous le nom "prodcartclient".

Remarque 1. Si cette opération ne semble pas avoir beaucoup de sens dans l'exemple précédent, ne pensez-vous pas qu'elle pourrait être utile pour interroger la base de données d'un site de rencontre ?

3 Sélection et projection

SQL
```
Rappel de la syntaxe SQL de base :

SELECT A1, A2, ...
FROM Table
WHERE Condition Booléenne de sélection

SELECT * FROM ...   signifie que l'on sélectionne TOUS les attributs de la table
```

A] SELECTION

Il s'agit ici de sélectionner des enregistrements répondant à un ou plusieurs critères dans une table donnée.
Nous allons ici travailler sur la table "client" qui réunit les deux tables "client1" et "client2" précédentes.

1. *Sélectionner toutes les femmes parmi les clients*
2. *Sélectionner tous les clients dont l'âge est inférieur à 30 ans*
3. *Sélectionner les hommes dont l'âge est supérieur à 50 ans.*
4. *Sélectionner les femmes qui ne peuvent pas faire partie de la population active.*

B] PROJECTION

Il s'agit ici de ne conserver que certains attributs parmi les attributs d'une table donnée.
Nous allons encore travailler sur la table "client" que nous avons conçue dans la partie précédente.

1. *Créer une requête permettant de ne conserver que les attributs "nomclient", "prenom" et "age" de la table.*

   Si l'on souhaite changer le nom des attributs, on peut ajouter "AS" dans la requête :

SQL
```
SELECT A1 AS newname1,
       A2 AS newname2, ...,
       AN AS newnameN
FROM Table
WHERE (condition booléenne)
```

   Modifier les noms de chaque attribut afin de tester cette fonctionnalité.

2. En général, les requêtes comportent à la fois :
   — des sélections : on ne souhaite conserver que les enregistrements vérifiant certains critères

— des projections : on ne s'intéresse qu'à certaines propriétés (attributs) de ces enregistrements

Créer une requête n'affichant que les nom, prénom et âge des clients dont l'âge est compris entre 30 et 50 ans.

Si l'on souhaite ordonner les enregistrements de la table obtenue, on peut ajouter "ORDER BY" dans la requête :

SQL
```
SELECT A1, A2, ...
FROM Table
WHERE (condition booléenne)
ORDER BY A1, A2                -- pour ordonner selon le couple (A1,A2)
```

Adapter la requête précédente afin que les enregistrements sélectionnés soient triés selon l'âge des clients.

3. Si cela n'a pas été fait, compléter la table "client" en y ajoutant un attribut "ville" et les données suivantes :

numclient	nomclient	prenom	age	sexe	ville
0	Hachard	Samuel	34	H	Paris
1	Berthy	Anissa	25	F	Londres
2	Allard	Kevin	45	H	Paris
3	Courcelle	Aline	71	F	Berlin
4	Kim	Sung-Ah	45	F	Londres
5	Ben chaid	Cherif	32	H	Madrid
6	Keita	Salif	13	H	Rome
7	Dilan	Bob	78	H	Londres
8	Gomez	Antonio	47	H	Rome
9	Garland	Todd	14	H	Paris
10	James	Tania	30	F	Berlin
11	Leer	Amanda	83	F	Londres
12	Pichon	Laure	24	F	Paris
13	Lin	Yu-Ting	29	F	Berlin
14	Berthy	Anissa	23	F	Berlin
15	Allard	Kevin	45	H	Paris

Construire une requête affichant les nom, prénom et âge des clients hommes habitant à Paris.

4 Jointure

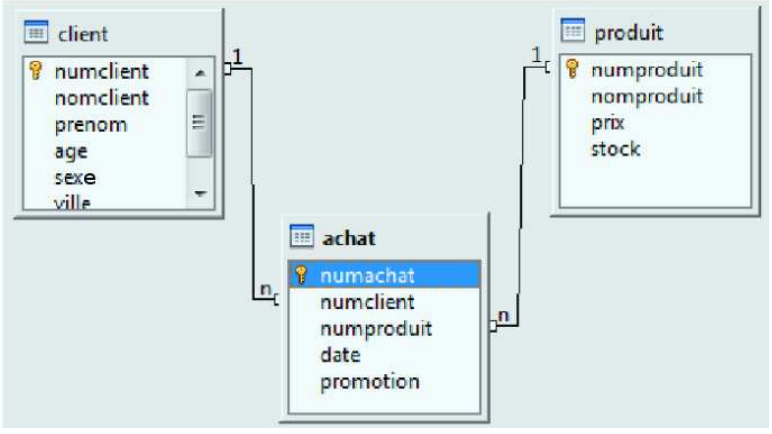
Nous allons dans cette section illustrer la notion de JOINTURE à l'aide des tables "client", "produit" et "achat" de notre base de données "fidelite.odb". Voici le contenu de la table "achat" :

numachat	numclient	numproduit	date	promotion
1	2	3	12/04/13	☑
2	15	4	02/01/13	☐
3	10	3	23/11/12	☑
4	2	1	12/04/13	☐
5	1	6	02/01/13	☑
6	5	2	11/12/12	☑
7	12	6	30/12/99	☐
8	11	4	12/11/12	☐
9	9	5	02/10/13	☐
10	10	2	12/09/12	☐
11	7	3	25/08/11	☑
12	8	1	21/10/11	☐
13	6	2	13/09/13	☐
14	14	4	04/05/12	☑
15	11	5	05/10/13	☐
16	7	3	12/12/12	☐
17	3	4	02/10/11	☐
18	4	1	12/11/12	☑
19	1	6	25/08/11	☐
20	2	2	17/02/11	☐
21	5	1	23/06/13	☐
22	3	3	26/03/12	☐
23	2	5	31/01/11	☐
24	13	2	22/05/12	☑
25	10	4	02/06/11	☐
26	11	6	05/10/12	☑
27	7	5	04/11/12	☐
28	3	1	07/05/11	☑
29	1	3	13/03/13	☐
30	5	1	27/04/11	☑

La table "achat" contient une clé primaire (numachat) et deux clés étrangères (numclient et numproduit) nous permettant d'effectuer une liaison avec les deux tables "client" et "produit".
Commençons par utiliser le mode EBAUCHE pour formuler notre première requête.
Les jointures avec le mode ébauche ne peuvent se faire qu'après avoir explicité les relations qui existent entre les trois tables. Nous allons procéder ainsi :

1. *Dans le menu "outil" sélectionner "relation..."*
2. *Ajouter dans la fenêtre qui s'ouvre les tables "client", "produit" et "achat"*
3. *Dans le menu "insertion" sélectionner "nouvelle relation..."*
   Sélectionner les champs qui relient les tables "client" et "achat" (ne rien modifier aux options).
4. *Renouveler l'opération précédente pour lier les tables "produit" et "achat".*

On obtient ainsi le "Modèle Relationnel de Données" suivant :



A] JOINTURE DIRECTE entre 2 tables :

Le fait que deux tables soient liées par un attribut commun nous autorise à effectuer des jointures entre elles.
A l'aide de l'option "créer une requête en mode ébauche", répondez aux questions suivantes :

1. *Les clientes Berthy et Kim achètent-t-elles essentiellement des produits en promotion ?*
   Pour obtenir cette information, nous avons besoin des tables "client" et "achat.

   On pourra constater que la traduction de la requête en langage SQL est :

   SQL
   ```
   SELECT client.nomclient, achat.promotion FROM achat, client
   WHERE     (client.nomclient = 'Berthy' OR  client.nomclient = 'Kim' )
         AND (achat.numclient = client.numclient)                          -- JOINTURE !!
   ORDER BY client.nomclient ASC, achat.promotion ASC
   ```

   Le codage SQL le plus efficace en terme de rapidité consiste à utiliser la commande `FROM R1 JOIN R2 ON ...`
   Ce qui donne :

   SQL
   ```
   SELECT client.nomclient, achat.promotion
   FROM achat JOIN client ON achat.numclient = client.numclient       -- JOINTURE !!
   WHERE   (client.nomclient = 'Berthy' OR  client.nomclient = 'Kim' )
   ORDER BY client.nomclient ASC, achat.promotion ASC
   ```

   MODIFIER la requête en utilisant l'instruction `FROM ... JOIN ... ON ...`.

   A partir de maintenant, vous tenterez de formuler directement vos requêtes en mode SQL en indiquant la jointure grâce à la commande `FROM... JOIN ... ON ...`!!

2. *Qui a acheté des produits le 12 avril 2013 ?*
   Pour obtenir cette information, nous avons besoin des tables "client" et "achat.
   En SQL, la date s'écrit {`D '2018-04-12'`}

3. *Quels produits ont été achetés le 12 avril 2013 ?*
   Pour obtenir cette information, nous avons besoin des tables "produit" et "achat".

4. *A quelles dates les clients masculins qui n'habitent pas Paris ont-ils effectué des achats ?*
   Pour obtenir cette information, nous avons besoin des tables "client" et "achat".

B] JOINTURE INDIRECTE entre deux tables :

Il est possible de regrouper des informations situées dans des tables liées entre elles par l'intermédiaire d'une troisième table. Dans notre exemple, les tables "client" et "produit" sont liées par l'intermédiaire de la table "achat" grâce aux attributs "numclient" (qui relie "client" à "achat") et "numproduit" (qui relie "produit" à "achat").

Ces deux jointures se formulent en SQL de la façon suivante :

SQL
```
SELECT ...
FROM achat JOIN client  ON achat.numclient = client.numclient      -- JOINTURE 1
           JOIN produit ON achat.numproduit = produit.numproduit   -- JOINTURE 2
WHERE ...
```

1. *Pouvez-vous dire ce qu'ont acheté les différents clients ?*
   Utiliser le résultat de la requête pour donner le nom des clients qui ont le plus consommé ?

2. *Donner la liste ordonnée des produits achetés par les clientes du magasin.*
   On affichera également les noms et prénoms des clientes.

3. *Donner la liste des produits achetés par les clients dont l'âge est compris entre 20 et 40 ans.*
   On ordonnera les résultats selon l'âge des clients.

5 Utilisation des fonctions d'agrégation

Les fonctions d'agrégation permettent d'effectuer des statistiques simples sur les données.

On peut ainsi par exemple :

1. Dénombrer des enregistrements
2. Calculer des sommes, le maximum ou le minimum d'un ensemble de valeurs
3. Calculer des moyennes, des écarts-types d'un ensemble de valeurs

Dans notre base de données "fidele.odb", nous disposons du champ "prix" de la table "produit" et du champ "age" de la table "client" sur lesquels il est possible de tester ces différentes fonctions.

Les requêtes seront directement rédigées en langage SQL.

A] Statistiques globales sur une table :

La syntaxe pour effectuer une statistique $f(A)$ sur un attribut A est la suivante :

SQL
```
SELECT f(A) FROM Table
```

1. Utilisation de la fonction `COUNT()` :

   (a) *Déterminer le nombre de produits listés (cad d'enregistrements) dans la table "produit".*
   Vérifiez que la table "produit" contient le même nombre de valeurs par colonne

2. Utilisation de la fonction `SUM()` :

   Déterminer la somme totale dépensée par les clients du magasin.

3. Utilisation de la fonction `AVG()` :

   Déterminer la moyenne des prix des produits contenus dans la table "produit".

4. Utilisation des fonctions `MAX(), MIN()` :

   Déterminer en une seule instruction :

   (a) Le nombre de prix,
   (b) La moyenne des prix,
   (c) La prix maximum,
   (d) Le prix minimum

B] Statistiques par catégories :

Dans cette partie, nous continuons à nous intéresser aux mêmes fonctions d'agrégation, mais cette fois par catégories.

Si, par exemple, on souhaite obtenir la moyenne de l'attribut A selon les différentes valeurs du couple (B1, B2), on utilisera l'instruction `GROUP BY` de la façon suivante :

SQL
```
SELECT B1, B2, AVG(A) FROM ...
GROUP BY B1,B2
```

1. *Déterminer la moyenne d'âge des clients par sexe.*

   Modifier les instructions SQL afin de donner un nom plus explicit à la moyenne des âges.

2. *Déterminer les sommes moyennes et les sommes totales dépensées par les hommes et les femmes. Quel est le groupe qui a dépensé le plus ?*

   Cette requête nécessite une jointure indirecte...

3. *Déterminer par ville d'origine, la moyenne d'âge des clients et la somme totale dépensée par ceux-ci.*

   Cette requête nécessite une jointure indirecte...

Remarque 2. Dans le prochain TD, nous verrons comment ajouter aux requêtes contenant des fonctions d'agrégation, des conditions de sélection en amont et en aval.

C] Requêtes diverses :

Exercice : 4

Requêtes simples :

1. *Classer les clients selon le nombre de produits achetés.*
2. *Quel est le meilleur client du supermarché ?*
3. *Existe-t-il des cases vides dans la colonne* `prix` *?*
4. *Combien d'argent les clients ont-ils dépensé en 2013 ?*
   On utilisera la commande `YEAR(achat.date) = 2013`.

Certaines requêtes nécessitent la formulation de requêtes préalables : on parle alors de "composition de requêtes". Pour formuler ces requêtes, comme il est d'usage en informatique, on commence par formuler les requêtes les plus simples, puis on les utilise (comme nouvelle table ou nouvelle valeur) dans les requêtes plus complexes.

Exercice : 5

Composition de requêtes :

1. *Déterminer les clients dont l'âge est supérieur à la moyenne des âges de tous les clients.*
2. *Déterminer les clients qui ont acheté plus de produits que la moyenne.*
3. *Déterminer la somme moyenne dépensée par client :* $S = \frac{somme\ totale\ dépensée}{nombre\ de\ clients}$