5 TP 5 : connexion à une base de données, classes intermédiaires

Utiliser les mécanismes vus dans le « TD 5 : relations à une base de données », soit en utilisant sqlite localement, soit via une liaison Oracle ou MySQL de l'UB.

Note : vous pouvez télécharger librement `SQLite` sur `https://sqlite.org/download.html`. Il existe des binaires précompilés pour votre système. Il faut les ajouter au `PATH` pour les utiliser avec `QT`.

5.1 Connexion à une base `SQLite`

Avec les sources fournies : `connexionBDD.zip`, vous avez un projet `QT` et une base de données `SQLite` très simples. Le projet `QT` déclare un widget sans interactions (c'est celui vu en CM) : à la construction du widget, dans la fonction `connexionBDD`, le fichier `/tmp/base.sqlite` est ouvert par le driver `QSQLDatabase` spécialisé pour `QSQLITE`. Une requête directe très simple est faite et les valeurs sont affichées.

La base de données est celle vue en cours :

```
idPersonne: INTEGER UNIQUE NOTNULL
nom:TEXT
prenom:TEXT
age:INTEGER
```

1. vérifiez que le programme se déroule correctement, que la requête s'exécute et que les données sont biens lues et affichées,
2. remplacez la requête directe par un `query.prepare()` (cf CM),
3. créez une requête qui ajoute un enregistrement, en utilisant un `prepare()` :

```
1 QSqlQuery query ;
2 query.prepare ("INSERT␣INTO␣personne␣(␣nom,␣prenom,␣age␣)␣VALUES␣(␣:n,␣:p,␣
    :a␣)");
3 query.bindValue (":n", "Etbianca");
4 query.bindValue (":p", "Bernard");
5 query.bindValue (":a", 12);
6 query.exec () ;
```

4. vérifiez qu'une suppression, avec un `prepare()` fonctionne également [1].

5.2 Application au projet

Selon votre schéma ou celui donné en TD2-3 (relations entre `Contact`, `Interaction`, `Todo`), créez une base de données multi-tables, remplissez-là de quelques valeurs (les fichiers CSV sur *plubel* peuvent vous y aider) et testez quelques requêtes avec une interface d'interrogation (« SQLiteDB-Browser », « dBeaver », « pgAdmin »...).

Utilisez les raccourcis de `QTCreator` : lors du remplissage d'un en-tête (`.h`) bouton droit sur une variable « Refactor / Generate getter and setters », bouton droit sur une fonction « Refactor / Ajouter une définition dans le .ccp ».

1. Des aides pour `SQL` : `https://www.sqltutorial.org/` et le *cheatsheet* `https://www.sqltutorial.org/wp-content/uploads/2016/04/SQL-cheat-sheet.pdf`

1. dans un widget QT (inutile de faire une `QMainWindow`, elle sera ré-écrite pour votre projet, autant faire un *composant*), créez une IHM qui permette de remplir les champs d'un `Contact` :

   ```
   nom
   entreprise
   photo (une chaîne de caractères pour le chemin d'accès)
   email (une chaîne également)
   date de création
   ```

2. créez (si ce n'est pas déjà fait pour votre projet) une classe `Contact` qui n'utilise pas de type de QT (il faut donc traduire ce qui vient de l'IHM), remplissez-en une instance lors de l'appui sur un bouton de ce formulaire,
3. une fois le contact créé, on demande son insertion dans la BDD. Écrire une classe intermédiaire comme `QSQLData` du TD 5 qui vous permettent d'émettre des signaux d'ordres vers la base de données (extraire la liste des contacts, insérer/supprimer un contact...)
4. procéder de la même façon (IHM, remplissage d'une structure C++ seulement, classe intermédiaire vers la BDD) pour les interactions :

   ```
   date //QDate pour l'instant
   contenu //QString pour l'instant
   ```

   Attention, une interaction doit être liée à un contact. Il faut donc un champ du formulaire de saisie qui soit une liste (par exemple déroulante). Vous utiliserez une `QComboBox` pour cela. Pour le test (c'est à dire sans faire de liaison encore avec la liste des contacts), on peut l'initialiser avec un `QStringList`, comme :

   ```
   1 lContacts = new QComboBox(); //déclaré dans le .h
   2
   3 QStringList contacts = (QStringList() << "jean␣Dupont" << "bernard␣EtBianca
       " << "georges␣Boy");
   4
   5 lContacts->addItems(contacts);
   ```

   Pour savoir quel élément a été sélectionné lors de la validation du formulaire (dans le slot qui reçoit le `clicked()` du `QPushButton`) :

   ```
   1 lContacts->currentText(); //retourne un QString
   ```

5. pour finir, au choix (dans le binôme de projet) :
   - inclure ce widget dans une `QMainWindow`. Pour le faire apparaître, on remplace le `central widget` :

     ```
     1 wAddContact = new widgetAddContact();
     2 setCentralWidget(wAddContact);
     ```

     L'idéal étant de le déclencher par une action du menu : modifier le `.ui` de la `MainWindow`, dans l'`Action Editor` de `QTDesigner` sélectionner l'action, bouton droit « Aller au slot ».
   - terminer la liaison `Contact`/`Interaction` en 1) important dans la `ComboBox` les noms des contacts (ou nom et entreprise), 2) dans le gestionnaire de BDD, récupérer l'`idContact` et l'`idInteraction` à l'insertion, et remplir la table d'association `ContactInteraction`. Vérifier avec une requête SQL que les associations sont faites.