Provide the bank data needed for financial applications in SQLite format

Synopsis:

KtoBLZCheck is a library to check bank account numbers and bank codes (BLZ) of German Banks and other international banks. It is based on the specifications of the "Deutsche Bundesbank".
Currently, KtoBLZCheck uses multiple bankdata files in text format which are valid at different dates. The data which is regularly retrieved from Deutsche Bundesbank is valid for three months. If the current text file is outdated at the time of usage a warning is printed and the user is requested to install an updated file. This leads to a lot of duplication of data.

The idea is to integrate the support for query and generation of SQLite databases into KtoBLZCheck to replace the text files used currently. This will avoid duplication of data and enable the support for multiple countries improving the scope of this library.
We also aim to create an API for querying the databases to integrate these databases into other applications.

Mentor: Ralf Habacker

Project Goals:

Here is a list of goals, that this project aims to meet:

- Replace the currently used text bankdata files with an SQLite database with support for validity period checking. The database should be generated and updated from the bank data retrieved from Deutsche Bundesbank.

- Make changes in the library to use the generated database as the bank data file in place of the text files.

- Make changes to the command-line tool to accept an SQLite database as an input data file from the user.

- Add support for additional countries to the checking library. The bankdata files also need to be added corresponding to these countries.

- Create an API for querying the SQLite databases to enable the integration of these databases into other applications.

- Write unit tests and documentation for the added code.

Implementation:

The first task is to generate the SQLite databases corresponding to the bankdata_*.txt files. The same can be done by this python script for the German banks. Some changes have to be made to the script to support validity period checking. Mainly, columns for valid_from and valid_upto dates have to be added. To generate the database at compile time from the data retrieved from Deutsche Bundesbank, I will modify this add_custom_command in CMakeLists.txt file in the src/bankdata/ directory to run this script at build time.

```
# convert into ktoblzcheck format
add_custom_command(
    OUTPUT ${BANKDATA_FILEPATH}
    COMMAND ${Python_EXECUTABLE} ${CMAKE_CURRENT_SOURCE_DIR}/bankdata.py -o ${BANKDATA_FILEPATH} ${BANKDATA_RAW_FILEPATH}
    DEPENDS ${BANKDATA_RAW_FILEPATH}
    WORKING_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}
)
```

Another task is to make changes to the command-line tool to use SQLite DB file as user input in place of text files. To implement this, the following code in the /src/bin/ktoblzcheck.cc file will be changed to use the APIs from the SQLite library. Although the tool will work only after the code in the library is modified to use the SQLite DB in place of text files.

```
//        }
        // No need to load the data file a second time - the default
        // constructor already loads the most suitable file for today.
    } else {
        ifstream test(bankdataFile.c_str());
        if (test.fail()) {
            std::cerr << PACKAGE ": Warning: Specified bankdata file '"
                      << bankdataFile
                      << "' could not be opened. Trying default file."
                      << std::endl;
            check_ptr = new AccountNumberCheck();
        } else {
            check_ptr = new AccountNumberCheck(bankdataFile);
        }
    }
    assert(check_ptr);
```

A lot of changes have to be made in the library code specifically in the class AccountNumberCheck. Many of the methods in the class do text file reading that has to be modified to use SQLite library APIs. I will also implement some tests for this changed code.

For support of additional countries, similar python scripts have to be used. Each additional country would be added as a separate SQLite database and changes in the library will be done to use them.

Next part would be to implement an API for the database. I would use a prior implementation in KMyMoney as a reference for this part. This will allow other applications to integrate these databases into their code.
At last, a lot of testing will be done to ensure everything works as intended. Detailed comments and updates in documentation will be done.

Timeline:

Duration	Deliverables
May 4, 2020 - June 1, 2020	**Community Bonding Period** In this period, I will spend my time going through the source code and getting familiar with it. I will discuss this project with my mentor and make changes or additions if required. At the end of this period, I aim to get a thorough plan of the implementation. I will also start a blog and add it to the KDE Planet for sharing my experiences on this journey.
June 1, 2020 - June 8, 2020	Generate the SQLite database corresponding to the bankdata files. I will have it automatically generated on build time. I will make changes in python scripts to add date columns.
June 8, 2020 - June 15, 2020	I will start working on replacing the current code with code working with SQLite database.

June 15, 2020 - June 22, 2020	I will work on replacing the current code with code working with SQLite database. I will update unit tests and documentation to reflect the above changes.
June 22, 2020 - June 29, 2020	I will modify the command-line tool to take SQLite DB file as input. I will refactor test cases and update documentation if needed. I will also write a blog on experiences of the first phase of GSoC.
June 29, 2020 - July 3, 2020	**Phase - 1 Evaluation**
July 4, 2020 - July 11, 2020	I will finish any pending work. I will integrate database generating scripts for additional countries into the code. I will also refactor/simplify the CMake build system.
July 11, 2020 - July 18, 2020	I will finish any pending work. I will make changes in code to support the additional countries SQLite databases. I will write tests and documentation for the extended code.

July 19, 2020 - July 26, 2020	I will finish any pending work. I will finish adding support for additional countries. I will then start work on the API. I will write another blog on the challenges and experiences of the second phase of GSoC.
July 27, 2020 - July 31, 2020	**Phase - 2 Evaluation**
August 1, 2020 - August 16, 2020	I will finish the API to allow integration of databases into other applications. These will be tested and verified for their expected behaviour and their actual behaviour. I will write documentation for this API.
August 17, 2020 - August 24, 2020	I will finish any pending work. I will also write a blog on the whole experience and journey of GSoC.
August 24, 2020 - August 31, 2020	**Phase - 3 Evaluation (Final)**
September 8, 2020	**Results Announced**

About Me:

I am Prasun Kumar, a student of Indian Institute of Information Technology, Guwahati (Assam, India). I am in my 2nd year (4th) semester of Bachelor in Technology in Computer Science and Engineering.

I have a working experience of working in C++ and C having used them for almost two years in my assignments, competitions and personal projects. I also have a working knowledge of Qt GUI development.

I am new to the KDE community and open source development having started just 3 months back but I am very excited to be a part of it. I have contributed to the KDE project KStars which is an astronomy software. I have worked on fixing a junior job which is in the following commit:
https://phabricator.kde.org/D27890

Since, KtoBLZCheck didn't have any open bugs or feature requests at the time, I wasn't able to contribute to this project.
To be a part of the FOSS development is a great honour for me and I would love to be a contributor to it.

From the past few months, I am actively participating in this project. I am able to learn new things at a fast pace.
I look forward to this project to learn more and more things and be an active part of the organisation.

Starting from May 4, 2020, I don't have any prior commitments and will be able to give 40 to 45 hours a week to this project. My college will reopen in August but still, I would be able to give about 20 to 25 hours per week.

I am not submitting proposals to any other organisation other than KDE.

Coding Skills:

- **Languages:** Proficient in C and C++. Also have a working knowledge of Python, Java and basic Javascript.

- **Databases:** MySQL, SQLite.

Coding Environment:

- **IDE:** Microsoft VSCode, Qt Creator.

- **Operating Systems:** Ubuntu 19.10 (preferred), KDE Neon (in VM), Windows 10.

Contact Details:

- **E-mail:** prasun.code@gmail.com
- **Github:** https://github.com/prasunka
- **LinkedIn:** https://www.linkedin.com/in/prasunkr/
- **Phone No.:** +91-8873741976