SQLite RDBMS Extension for Data Indexing Using B-tree Modifications

The world processed data amount increases continuously. Doubling of the transistor-based CPUs performance (according to the Moore’s law) approximately every two years does not cover the data amount growth. The world faces the big data problem.

Many companies and laboratories need to gather, store and process big data. There are many algorithmic and software solutions for these problems, however the most voluminous data storages are the slowest. This problem is partially solved by using indexes, which are usually represented by data structures such as hash tables and trees.

In addition, when data are stored on slow carriers it is more efficient to load data batches from a storage and not separate data elements. Multiway trees solve this problem. The most widespread kinds of multiway trees are B-tree and its modifications, B^+-tree and B^*-tree. However, the popular open-source embedded relational DBMS SQLite does not support using B^+-tree or B^*-tree as data indexes.

The main goals of the work are to add B-tree modifications such as B^+-tree, B^*-tree and B^{*+}-tree (the latter was developed by the author of this work data structure, which combines the main B^+-tree and B^*-tree features) to SQLite and to develop an algorithm that would allow selecting the most appropriate indexing data structure (B-tree, B^+-tree, B^*-tree or B^{*+}-tree) when a user creates a table.

The work includes linking of B-tree modifications to a C++ library (developed by the author of this work) to SQLite using a C-C++ cross-language API, developing an algorithm for selecting an indexing data structure and measuring indexing efficiency parameters’ values (indexing and searching time, memory usage and disk operations count).