Исследованиеподдержанопроектом CERES. Centers of Excellence for young RESearchers (Reg.no. 544137-TEMPUS-1-2013-SK-JPHES),

Список использованных источников:
1. Daryl L. Logan. A first course in the finite element method – Cengage Learning, 2011. – 752 с.
2. О.Зенкевич. Метод конечных элементов в технике – Москва, 1975. – 541 с.
3. Метод конечных элементов [Электронный ресурс], https://ru.wikipedia.org/wiki/Метод_конечных_элементов

ПОЛНОТЕКСТОВЫЙ ПОИСК В ANDROID ПРИЛОЖЕНИЯХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вашило К. Р.

Хотеев А.Л. – канд. физ.-мат. наук, доцент

Рассмотрены ключевые аспекты организации полнотекстового поиска в приложениях ОС Android. Приведены результаты сравнения поиска в обычных таблицах SQLite и таблицах с настроенной поддержкой полнотекстового поиска.

Как только в мобильном приложении приходится хранить достаточно много информации, возникает проблема скорости поиска и доступа к интересующим пользователя данным. Типы приложений при этом могут быть совершенно различными, следует перечислить основные источники больших объемов текстовой информации: сообщения и переписка, текстовые документы, справочные материалы, книги. Следует учесть, что далее речь идет о поиске по документам, находящимся именно на телефоне (носимом устройстве Android) пользователя, где объем данных ограничен, как и мощности существующих инструментов. Если требуется работа на больших объёмах данных и сложных запросах (единицы и десятки Гб текста одного документа/записи), следует реализовать собственную поисковую систему на стороне сервера.

Первые реализации полнотекстового поиска сканировали содержимое всех документов для нахождения заданной для поиска фразы. При этом время поиска было неприемлемо велико и зависело от размера базы данных. Следующим шагом было создание полнотекстового индекса - словаря, в котором перечислены все слова или термы текста и места, где они находятся в тексте.

В качестве базы данных на OS Android используется SQLite. Модули FTS3 и FTS4 расширяют SQLite и делают возможным создание специальных виртуальных таблиц, поддерживающих полнотекстовый индекс. Модули FTS1, FTS2 считаются устаревшими, и из-за известных проблем их использование нежелательно.

Сравнение использования FTS и запроса SQL. Добавим набор данных "Энроновская коллекцияe-mail“ из более пятиста тысяч документов в таблицу FTS и обычную таблицу SQLite.

```
CREATE TABLE data1(body TEXT);
CREATE VIRTUAL TABLE data2 USING fts3(body TEXT);
```

Листинг 1 – Пример создания обычной таблицы SQLite и аналогичной таблицы FTS

Проанализируем результаты работы приведенных запросов к базе данных:

```
SELECT count(*) FROM data1 WHERE body MATCH 'android';  /* результат 22.5 секунды */
SELECT count(*) FROM data2 WHERE body LIKE ‘%android%'; /* результат 0.03 секунды */
```

Листинг 2 – Пример выполнения запросов обычной таблице SQLite и в таблице FTS

Оба поисковых запроса являются регистронезависимыми. На заполнение данными таблицы FTS3 ушло чуть больше времени: 30 и 25 минут. Однако следует отменить, что приведенные запросы не являются эквивалентными. Перечислим основные различия работы запросов:

1) Результаты запроса LIKE могут включать в себя слова "androidphobe" или “FlyAndroid”. Запрос MATCH отдает результатом только строки, содержащие "android" как отдельное вхождение.

2) Для хранения таблицы FTS3 понадобилось на треть больше дискового пространства (~2 Гб).

С точки зрения разработчика таблицы FTS в целом схожи с обычными таблицами SQLite. К ним применимы те же команды INSERT, UPDATE, DELETE для добавления, редактирования и удаления данных. Для запроса данных используется SELECT. Чтобы сделать запрос, задействуя встроенный полнотекстовый индекс, используется оператор MATCH.

Различия между FTS3 и FTS4. FTS4 является усовершенствованным FTS3. FTS4 не поддерживается в

Android до версии 3.0 (версия API 11). Если в приложении требуется поддержка ранних версий Android, то есть возможность добавить код работы с FTS3 таблицами только для версий Android до одиннадцатой, так как программные интерфейсы FTS3 и FTS4 во многом идентичны.

Токенайзеры. FTS Токенайзер — алгоритм извлечения термов из текста. Токенайзер задействуется как при индексации текста документов во время добавления их в базу данных, так и при осуществлении запроса к базе данных (над текстом запроса). По умолчанию используется “simple” токенайзер. Основные положения алгоритма “simple” токенайзера:

Термом считается последовательность из UTF-символов, код которых больше или равен 128, из символов букв, цифр, символа “_”. Все остальные символы отбрасываются.

Все символы верхнего регистра набора ASCII преобразуются в свои эквиваленты нижнего регистра.

Кпримеру, применималгоритм “simple” ктексту “This time we're really breaking up”. Мыполучимследующиетермы: “this time we re really breaking up”. Термы будут добавлены в полнотекстовый индекс и данная фраза будет найдена по такому запросу, как “MATCH ‘Really’”, ведь благодаря “simple” токенайзеру полнотекстовые запросы являются регистронезависимыми.

Наряду с “simple” существуют и другие алгоритмы, например “porter” токенайзер на базе алгоритма стеминга Портера. Стемминг - это нахождение основы слова, которая не всегда совпадает с морфологическим корнем слова. К примеру “porter” включает в себя правила удаления окончаний слов: “ed”, “ing”, “ly”, а перед выделением основ термов проводится “simple” токинезация. Результат обрабатки того же документа, что и в предыдущем абзаце: "thi time we re realli break up".

Таким образом, использование функциональности полнотекстового поиска чаще всего дает значительный выигрыш во в ремени, потраченном на нахождение результатов в больших объемах данных, позволяет довольно легко реализовать вывод результатов поиска в виде списка фрагментов документов. Данный подход должен быть применен во всех приложениях, в которых есть функционал поиска по относительно большим объемам данных, чтобы обеспечить положительный пользовательский опыт при работе с Android приложением.

Список использованных источников:

1. Википедия. Полнотекстовый поиск [Электронный ресурс] – Режим доступа. – URLhttps://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D0%BE%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA (дата обращения 25.03.2017).
2. Adding Search Functionality . Storing and Searching for Data [Электронный ресурс] – Режим доступа. – URL https://developer.android.com/training/search/search.html (дата обращения 25.03.2017).
3. SQLite FTS3 and FTS4 Extensions [Электронный ресурс] – Режим доступа. – URL http://www.sqlite.org/fts3.html (дата обращения 25.03.2017).

ПРИМЕНЕНИЕ ФИЛЬТРА КАЛМАНА В ОБРАБОТКЕ ИЗОБРАЖЕНИЙ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Галдунов Д.В.

Стройникова Е.Д. – ассистент кафедры информатики

В настоящее время проводятся множество исследований в области обработки изображений и компьютерного зрения, и результаты этих исследований активно используются в научной, производственной и других сферах деятельности. Методы обработки графической информации чаще всего применяются на начальных этапах работы алгоритмов и ставят своей целью очистку данных для дальнейших вычислений. В качестве одного из таких методов может выступать метод калмановской фильтрации, впервые опубликованный в 1960 г.

Фильтр Калмана – линейный рекурсивный фильтр, который используется для получения оценки состояния динамической системы по серии неточных измерений. Модель системы представлена уравнением

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1},$$

где вектор x_i – истинное состояние системы в момент времени *i*; матрица *A* порядка $n \times n$ связывает состояние в предыдущий момент времени $k-1$ с состоянием в текущей момент *k*; $n \times l$ матрица *B* связывает опциональный входной вектор управляющих воздействий *u* с вектором *x*; w_i является случайным вектором. Измеренное состояние системы описывается уравнением

$$z_k = Hx_k + v_k,$$

где $m \times n$ матрица *H* связывает состояние *x* с измерением *z*; случайный вектор v_i представляет собой ошибку измерений прибора. Предполагается, что векторы *w* и *v* являются взаимно независимыми и распределенными нормально с нулевым математическим ожиданием и ковариационными матрицами *Q* и *R* соответственно.

В области обработки графических данных фильтр Калмана может быть использован для решения