УДК 004.42

АВТОМАТИЗИРОВАННЫЙ ЧУВАШСКО-РУССКИЙ ПЕРЕВОДЧИК: РЕАЛИЗАЦИЯ С ПОМОЩЬЮ МЕТОДА ПРОГРАММНОЙ СИМУЛЯЦИИ SQL-ЗАПРОСОВ ТИПА SELECT

Димитриев А.П., Лавина Т.А.

ФГБОУ ВО «Чувашский государственный университет имени И.Н. Ульянова», Чебоксары, e-mail: dimitrie1@yandex.ru

Предметом исследования является разработанный авторами данной статьи программный комплекс, предназначенный для автоматизированного перевода с чувашского языка на русский язык. В предыдущей версии этого программного комплекса выявлены определенные недостатки. Это необходимость установки на компьютер дополнительного программного обеспечения, низкая скорость обработки текста и опасность износа применяемой внешней памяти. Причиной некоторых из этих недостатков являлось интенсивное использование внешней памяти в режиме записи. Статья посвящена устранению выявленных недочетов. В результате устранения этих недостатков скорость обработки текста значительно возросла, а также снижена нагрузка на физический носитель данных. Кроме этого, теперь нет необходимости в установке на компьютер дополнительного программного обеспечения: требуется только операционная система Windows. Таких результатов удалось достичь за счет несколько усложненного программирования, выражающегося в программной симуляции SQL-запросов к базе данных. При этом используются несколько массивов, в которые загружаются все данные, которые ранее хранились в базе данных. Это приводит к увеличению требуемого объема используемой оперативной памяти, но для конечного пользователя это может считаться несущественным. Приводится словесное описание алгоритмов, реализующих поиск в этих массивах. Новые алгоритмы, аналогично алгоритмам, использующим индексы в реляционных базах данных, обладают временной сложностью, пропорциональной логарифму от количества записей массива.

Ключевые слова: оператор SELECT, автоматизированный перевод, чувашский язык, отбор данных, база данных

AUTOMATED CHUVASH-RUSSIAN TRANSLATOR: IMPLEMENTATION USING THE METHOD OF SOFTWARE SIMULATION OF SELECT-TYPE SQL QUERIES

Dimitriev A.P., Lavina T.A.

Chuvash State University named after I.N. Ulyanov, Cheboksary, e-mail: dimitrie1 @yandex.ru

The subject of the study is a software package developed by the authors of this article, designed for automated translation from the Chuvash language into Russian. In the previous version of this software package, certain shortcomings were identified. This is the need to install additional software on the computer, low text processing speed and the risk of wear of the external memory used. The reason for some of these shortcomings was the intensive use of external memory in write mode. The article is devoted to the elimination of the identified shortcomings. As a result of eliminating these shortcomings, the speed of text processing has increased significantly, and the load on the physical data carrier has also been reduced. In addition, now there is no need to install additional software on the computer: only the Windows operating system is required. Such results were achieved due to somewhat complicated programming, expressed in software simulation of SQL queries to the database. This uses several arrays into which all the data that was previously stored in the database is loaded. This leads to an increase in the required amount of RAM used, but for the end user this may be considered insignificant. A verbal description of the algorithms that implement the search in these arrays is given. Algorithms, like algorithms using indexes in relational databases, have time complexity proportional to the logarithm of the number of array entries.

Keywords: SELECT operator, computer-aided translation, Chuvash language, data selection, database

Чувашский язык нуждается в поддержке, так как, по данным ЮНЕСКО, относится к определенно находящимся под угрозой исчезновения («входит в список вымирающих») [1, с. 37], что отображается в [2]. Ввиду повсеместного внедрения информационных технологий реализация поддержки со стороны таких технологий является актуальной. В связи с этим различные разработчики прикладывают свои усилия. Например, появился бесплатный сервис чувашско-русского автоматизированного статистического перевода в Яндексе, однако сторонние разработчики не имеют доступа к его API и базе данных. Актуальность внедрения информационных технологий для чувашского языка дополнительно подтверждается появлением переводчика с чувашского языка на татарский в составе платформы OpenTrad Apertium [3, с. 171], [4], а также разработкой Национального корпуса чувашского языка [5, 6]. На кафедре компьютерных технологий Чувашского государственного университета также разработан автоматизированный чувашско-русский переводчик, реализованный в качестве приложения Windows [7]. Однако он обладал некоторыми недостатками, такими как: необходимость в дополнительном программном обеспечении, низкая про-

изводительность, интенсивное перезаписывание данных на носителе информации.

Целью работы является преодоление перечисленных недостатков автоматизированного чувашско-русского переводчика. Решаемые задачи: проанализировать эти недостатки, выполнить программную симуляцию SQL-запросов, протестировать работоспособность программы.

Материал и методы исследования

Исследуется программный комплекс для автоматизированного чувашско-русского перевода, разработанный в Чувашском государственном университете и представляющий собой приложение Windows. Для исследования используется программирование в интегрированной среде Turbo Delphi, в которой данный программный комплекс был разработан. Рассуждения подтверждаются ссылками на литературу и электронные источники на официальных сайтах. Числовые результаты усредняются по результатам серий испытаний.

Результаты исследования и их обсуждение

Как уже отмечено, одним из недостатков является необходимость установки на компьютер дополнительного программного обеспечения, а именно Borland Database Engine (BDE). Оно доступно через сеть Интернет по различным ссылкам, однако у пользователей эти ссылки могут не вызывать доверия, в частности потому что самой компании Borland – разработчика – не существует уже более 10 лет. С сайта Управления информатизации Чувашского государственного университета [8] можно загрузить интегрированную среду Turbo Delphi, затем ее установить, и на компьютере будет попутно установлено BDE. В данном случае вредоносных функций у данного программного обеспечения гарантированно нет. Однако для установки требуются права администратора, которые не всегда имеются (например, в компьютерных классах коллективного пользования), а при их наличии нужно некоторое время и принятие ряда решений при установке. Кроме того, в случае установки Turbo Delphi будет использована часть системных ресурсов: внешняя память для хранения файлов, визуальное пространство для отображения ярлыков, процессорное время при периодической проверке компьютера на вирусы.

Следующим недостатком является низкая скорость обработки текста. Для сравнения: если применять «Яндекс Переводчик» для автоматизированного перевода с английского языка на русский, то перевод 1000 знаков занимает менее секунды, включая время передачи по сети, что выявлено методом усреднения измеренного времени по 10 испытаниям. Однако перевод предложения всего из 10 слов среднего размера с чувашского языка на русский в разработанной авторами с участием других лиц программе занимает в среднем около 10 секунд, что во много раз ниже привычной скорости перевода. Значительная часть времени при этом затрачивается на морфологический анализ. В направлении ускорения морфологического анализа слов на чувашском языке значительные успехи достигнуты в [9, с. 17]: 10 слов за 7 миллисекунд. Следовательно, скорость выполнения морфологического анализа в исследуемом программном комплексе также может быть увеличена.

Третий недостаток связан с применяемым способом обработки текста, основанным на интенсивном использовании внешней памяти в режиме не только для чтения, но и для записи. Это вызывает опасения по поводу преждевременного износа применяемой памяти, которые особенно актуальны при применении флеш-накопителя. Известно, что количество циклов программирования-стирания, которые может выдержать ячейка флеш-памяти до износа, составляет от 1000 до 100 000, что зависит от типа флеш-памяти [10]. Таким образом, если по невнимательности или незнанию запустить программу непосредственно с флеш-накопителя, неизбежен его износ в той или иной степени, что зависит от объема переводимого текста. Жесткие диски во время работы также подвержены износу, но в основном по другим причинам (например, при ударах), а при интенсивной загрузке жесткого диска, например антивирусом, дополнительная нагрузка на него со стороны программы-переводчика приводит к увеличению дискомфорта пользователя при работе с другими программами, запущенными одновременно и требующими обращений к жесткому диску.

Низкая скорость работы переводчика объяснялась созданием многочисленных временных файлов с результатами многократных SQL-запросов для каждого переводимого слова. Для преодоления данного фактора замедления вначале исследована замена всех SQL-запросов на применение фильтров в свойствах таблиц. Но скорость работы все еще оставалась низкой – на уровне, сопоставимом с изначальным. Поэтому предлагается использовать программную симуляцию SQL-запросов с помощью организации поиска в массивах записей, хранящихся в оперативной памяти, следующим образом.

При запуске программы производится считывание всех необходимых данных из текстовых файлов в указанные массивы (10 двумерных массивов – таблиц, включающих от 100 до 175 000 строк и 2–6 столбцов). Это приводит к увеличению требуемого объема оперативной памяти, но для современных компьютеров оно несущественно. Так, в Windows 7 объем памяти, занимаемой программой после загрузки указанных массивов, составляет 141 Мбайт.

Эти данные ранее образовывали базу данных, состоящую из файлов формата Dbase, индексных файлов и файлов с информацией об индексах – итого по три файла на каждую таблицу. Теперь индексная информация отсутствует, вместо этого данные физически упорядочены в текстовых файлах. Всего используется 10 файлов общим объемом 136 Мбайт, тогда как ранее было 48 файлов с суммарным объемом 173 Мбайт. Ранее для работы с базой данных требовалась установка BDE, а теперь нет такой необходимости. Однако BDE позволяло просматривать базу данных в интерактивном режиме в виде таблиц, а в новой версии такая возможность исключена. Однако для конечного пользователя в этом нет необходимости.

Для реализации поиска и последующего отбора данных из массивов (т.е. симуляции оператора SELECT языка SQL) используется поиск по бинарному дереву. Сортировка в большинстве таблиц произведена только по одному столбцу, так как для решаемой задачи большего количества обычно не требуется.

После того как в некоторой записи (фактически строке) массива будет найдено одно из значений, удовлетворяющих условию поиска, производится последовательный анализ расположенных рядом сверху и снизу строк массива на предмет удовлетворения тому же условию. Когда условие перестает удовлетворяться, продвижение вверх или вниз по строкам массива прекращается.

Результаты такого отбора помещаются в специальный результирующий массив записей типа *tqr* (см. приводимый ниже листинг), ограниченный по количеству строк (не более 100, так как больше не требуется). Записи этого массива предназначены для хранения данных из пяти тех или иных столбцов таблицы, к которой применяется отбор (больше в программе не требуется). Тип данных каждого поля – строка длиной от 55 до 70 символов, что определяется как максимум от необходимой длины. Данные в результирующем массиве далее используются таким же образом, как и ранее в результирующем наборе оператора SELECT.

После программной реализации данного подхода проведено функциональное тестирование на основе структуры (т.е. методом «белого ящика») [11, с. 4] и выявлены следующие проблемы.

1. В чувашском языке имеются буквы, которые представляются в некоторых специальных шрифтах для чувашского языка, например Times ET Chuvas, символами «ё», точкой, слешем и т.п. Это препятствует правильной сортировке данных привычными методами, например с помощью СУБД Microsoft Access, а также бинарному поиску, который производится по кодам символов. Для решения данной проблемы учитывалось, что бинарный поиск реализован в Turbo Delphi, поэтому средствами Delphi осуществлена «правильная» по отношению к специфическим чувашским буквам сортировка текстового файла, полученного с помощью Access из файлов формата Dbase. После сортировки полученные данные сохранены в новый текстовый файл.

2. Один из файлов содержит данные о статистике появления русских словосочетаний. В русских словах встречается буква «ё», и в обоих регистрах код символа этой буквы находится не рядом с кодом символа буквы «е», причем в текстах нередко вместо «ё» пишут «е». В совокупности это вызывало ошибки при реализации двоичного поиска в Delphi. Поэтому потребовалось пересортировать этот файл, также средствами Delphi. Данные в этом файле являются ссылками на абсолютный номер записи в базе данных. В связи с пересортировкой значения указанных ссылок оказываются неверными. Для преодоления указанной проблемы введены столбцы, указывающие, под каким номером строки текущая строка была ранее и на какую строку теперь надо ссылаться, когда ссылаются на эту строку. Для этого в тип, описывающий запись строки, добавлены поля с именами *gdebyl* и *kudasm* типа Longint, как видно из фрагмента листинга, описывающего типы:

```
t200=array[1..200]of byte;
tqr=record f1:string[55]; f2:string[65]; f3:string[70];f4,f5:string[55] end;
TMRon2 = record F0:string[23];f1,f2,f3:t200;gdebyl,kudasm:longint;end;
TMAon2 = array[1..175000] of TMRon2;PMAon2=^TMAon2;
```

В данном листинге описываются типы, применяемые для хранения массива с данными о статистике появления словосочетаний объемом около 110 Мбайт, содержащего 175 000 строк. Ранее вместо такого массива использовалась таблица базы данных Dbase с индексом. Кроме того, при преобразовании соответствующего файла базы данных в текст двоичные данные иногда интерпретировались как начало новой записи, это вызывало сбой нумерации, что исправлялось вручную.

3. Отбор может осуществляться не только по условию равенства значения элемента массива искомому значению, но и по условию равенства его начала этому значению, а также это относится к сравнению двух значений элементов с двумя искомыми значениями (аналог отбора по значениям двух полей записи в таблице). Это реализовать несколько сложнее, чем в случае сравнения на совпадение с одним значением. Для реализации такого отбора использован следующий подход. Условие равенства начала строки программируется с использованием функции копирования начала строки, а сравнение с двумя значениями реализуется путем изначальной сортировки текстового файла с учетом значений двух полей.

Для осуществления данного подхода разработаны соответствующие алгоритмы. Программная реализация алгоритмов выполнена в виде функции *loc*2, осуществляющей поиск какого-либо совпадения строки, обозначаемой параметром функции *A*, со значениями, хранящимися в указанном параметром *T* глобальном массиве. Функция возвращает значение, интерпретируемое как «Найдено» или «Не найдено». Еще одним параметром функции является переменная *C* типа «Байт», которая сообщает, должно ли быть реализовано условие полного совпадения строк или совпадения первых символов в количестве, определяемом длиной *A*. Внутри функции объявлены следующие переменные:

1) *b* для хранения информации об успешности поиска, принимающая значения «Найдено» и «Не найдено»;

2) *i* – номер элемента массива;

3) *H*, *L* – соответственно текущие верхняя и нижняя границы сужающейся области поиска;

4) *s* – вспомогательная строка, в которую помещаются данные из текущей записи массива;

5) *G* – длина искомой строки.

Используются также глобальные переменные, являющиеся указателями на текущую запись заданного массива. Для каждого массива это свой указатель, например для массива *oon* переменная названа *oon_cr*, для массива *rch* – *rch_cr*.

Функция *loc*2 сравнивает значение *T* с константными строками в своем теле (именами таблиц) и при нахождении совпадения реализует соответствующий алгоритм поиска. Эти алгоритмы во многом сходны, отличия состоят в реализации или нереализации поиска по двум полям или по совпадению части строки. Один из этих алгоритмов, применяемый к массиву *oon* и реализующий поиск по двум полям, представлен далее.

1. Установить значения переменных: *b*← «Не найдено»; *L*←1; *H*← количество элементов массива; *j*←(*L* + *H*)*div*2, *G*← длина *A*.

2. До тех пор, пока выполняется условие $L \le H$, выполнять:

2.1. Если *C* = 2, то присвоить *s*← конкатенация первого поля и символьного представления второго поля *j*-й записи массива, иначе присвоить *s*← первое поле *j*-й записи массива без концевых пробелов.

2.2. Если *s* = *A*, то присвоить *b*← «Найдено» и перейти к шагу 3.

2.3. Если *s* > *A*, то присвоить $H \leftarrow j - 1$, иначе, если *s* < *A*, то присвоить $L \leftarrow j + 1$.

2.4. Присвоить $j \leftarrow (L + H)div2$.

3. Присвоить $i \leftarrow j - 1$.

4. Если *C* = 2, то выполнять:

4.1. До тех пор, пока *i* > 0 и конкатенация первого поля и символьного представления второго поля *i*-й записи массива равна *A*, выполнять:

4.1.1. Присвоить $j \leftarrow i$; $i \leftarrow i - 1$.

4.2. Перейти к шагу 6.

5. Если условие, проверяемое в шаге 4, не выполняется, то выполнять:

5.1. До тех пор, пока *i* > 0 и первое поле *i*-й записи массива без концевых пробелов равно *A*, выполнять:

5.1.1. Присвоить $j \leftarrow i$; $i \leftarrow i - 1$.

6. Присвоить глобальной переменной *oon_cr* ← *j*, возвращаемому функцией значению *loc*2 ← *b*.

В данном алгоритме на шаге 2.4 производится деление пополам множества записей, среди которых производится поиск значения. Этот шаг выполняется несколько раз, пока не будет найдена запись или делить будет нечего. Такая стратегия приводит к тому, что алгоритм обладает временной сложностью, пропорциональной двоичному логарифму от числа записей, т.е. $O(\log(H))$ при условии, что будет найдено немного записей. В наихудшем случае, когда будут найдены все записи, временная сложность составит $O(H)$.

Следующий алгоритм реализует поиск в массиве *rch* по совпадению части строки.

1. Установить значения переменных: *b*← «Не найдено»; *L*←1; *H*← количество элементов массива; *j*←(*L* + *H*)*div*2, *G*← длина *A*.

Среднее время обработки текста до и после доработки программы

Компьютер	t_1	$t_{1,m}$	$t_{1,x}$	$t_{1,mx}$	t_2	$t_{2,m}$	$t_{2,x}$	$t_{2,mx}$
1	114,7	116,7	155,3	153,3	21,2	21,5	23,5	23,1
2	26,8	28,7	33,6	34,6	5,2	5,4	5,6	5,6

2. До тех пор, пока выполняется условие $L \le H$, выполнять:

2.1. Если $C = 0$, то присвоить $s \leftarrow$ поле j-й записи массива без концевых пробелов, иначе присвоить $s \leftarrow$ первые G символов поля j-й записи массива без концевых пробелов.

2.2. Если $s = A$, то присвоить $b \leftarrow$ «Найдено» и перейти к шагу 3.

2.3. Если $s > A$, то присвоить $H \leftarrow j - 1$, иначе, если $s < A$, то присвоить $L \leftarrow j + 1$.

2.4. Присвоить $j \leftarrow (L + H)div2$.

3. Присвоить глобальной переменной $rch\_cr \leftarrow j$, возвращаемому значению $loc2 \leftarrow b$.

Выполнено тестирование производительности до и после доработки программы с использованием 20 файлов размером от 992 до 1003 символов с пробелами. Эти файлы содержат фрагменты текстов на чувашском языке, полученных с [6]. Файлы в среднем состоят из 136 слов, образующих 11 предложений. Вычисления производились на двух компьютерах: 1) ноутбук AMD E2-6110 APU 1500 МГц; 2) настольный компьютер Intel(R) Core(TM) i3-2120 3300 МГц. Введены следующие обозначения времени вычислений:

t_1, t_2 – для старой и новой версии программы с учетом только морфологии;

$t_{1,m}$, $t_{2,m}$ – с использованием статистики соответственно для старой и новой версии;

$t_{1,x}$, $t_{2,x}$ – при использовании синтаксического анализа для старой и новой версии;

$t_{1,mx}$, $t_{2,mx}$ – с применением как статистики, так и синтаксического анализа для старой и новой версии соответственно.

Результаты сведены в таблицу, где время указано в секундах.

Среднеквадратическое отклонение данных составляет от 7% до 23%. Как следует из таблицы, разница во времени для старой и новой версии может достигать десятков раз, а на одном и том же компьютере – в среднем 4,2 раза. Подобное увеличение скорости можно считать значительным успехом.

Заключение

Применение симуляции SQL-запросов с помощью массивов, несмотря на возросшие трудозатраты по программированию и переформированию данных, значительно ускоряет работу программы. Однако начальный запуск программы теперь замедлен приблизительно на 10 с, в зависимости от характеристик производительности применяемого компьютера, так как производится загрузка данных из текстовых файлов в массивы. Новая версия данной программы не требует установки никакого программного обеспечения: достаточно скопировать программный комплекс на компьютер, и он будет работоспособен. Добавление новых строк в таблицы требует ее физического переупорядочения, однако и ранее, при использовании файлов Dbase, это требовало некоторых действий, таких как перестроение индекса.

Список литературы

1. Желтов В.П., Желтов П.В. Разработка системы машинного перевода с чувашского на русский язык // Современные наукоемкие технологии. 2020. № 12-1. С. 37-42. DOI 10.17513/snt.38408.

2. UNESCO Atlas of the World's Languages in Danger. [Электронный ресурс]. URL: http://www.unesco.org/languages-atlas/index.php (дата обращения: 12.05.2022).

3. Желтов В.П., Желтов П.В. Создание двуязычного словаря для системы машинного перевода чувашского языка // Роль государства и институтов гражданского общества в сохранении родных языков и литератур: материалы Международной научно-практической конференции (Чебоксары, 22–23 октября 2020 г.). Чебоксары: ИД «Среда», 2021. С. 170–175. DOI 10.31483/a-10228.

4. Apertium Turkic. Открытая платформа машинного перевода. [Электронный ресурс]. URL: https://turkic.apertium.org (дата обращения: 12.05.2022).

5. Желтов В.П., Желтов П.В. Анализ национальных корпусов и новые возможности в изучении родных языков // Роль государства и институтов гражданского общества в сохранении родных языков и литератур: материалы Международной научно-практической конференции. Чебоксары: ИД «Среда», 2021. – С. 165–170. DOI 10.31483/a-10228.

6. Двуязычный корпус чувашского языка. [Электронный ресурс]. URL: https://ru.corpus.chv.su/ (дата обращения: 12.05.2022).

7. Димитриев А.П. Чувашско-русский переводчик: программная реализация // Прикладная информатика. 2011. № 6 (36). С. 43-46.

8. ФГБОУ ВО «Чувашский государственный университет им.И.Н. Ульянова». Управление информатизации. [Электронный ресурс]. URL: http://ui.chuvsu.ru/index.php/2010-06-25-10-45-35 (дата обращения: 12.05.2022).

9. Желтов В.П., Желтов П.В., Сергеев Е.С. Результаты тестирования программного обеспечения Национального корпуса чувашского языка // Современные наукоемкие технологии. 2017. № 8. С. 13-18.

10. Kingston Technology. Различия между типами памяти SLC, MLC, TLC и 3D NAND в USB-накопителях, твердотельных накопителях и картах памяти. [Электронный ресурс].; URL: https://www.kingston.com/ru/solutions/pc-performance/difference-between-slc-mlc-tlc-3d-nand (дата обращения: 12.05.2022).

11. ГОСТ Р 56920-2016/ISO/IEC/IEEE 29119-1:2013. Системная и программная инженерия. Тестирование программного обеспечения. Часть 1. Понятия и определения. М.: Стандартинформ, 2016. 54 с.