

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ

Заведующий кафедрой

_____/В.В. Шайдуров

(подпись) инициалы фамилия

« ____ » _____ 2023 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Расширение функционала базы данных для встроенных приложений

Направление 02.04.01 Математика и компьютерные науки

Магистерская программа 02.04.01.01 Математическое и компьютерное моделирование

Руководитель	доцент, кандидат физико-математических наук	С.Н. Баранов
Выпускник		И.Б. Смирнов
Нормоконтролер	доцент, кандидат физико-математических наук	Т.Н. Шипина

Красноярск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1. Конструкторский раздел	7
1.1 Постановка задачи	7
1.2 RTEMS	7
1.3 Real-time operating system (RTOS)	7
2. История	9
2.1 Описание основ	9
2.2 SQLite	12
3. Обзор LittleD	14
3.1 Перевод запросов	15
3.2 Распределение памяти	17
3.3 Индексирование	20
3.4 Внесение нового функционала	21
ЗАКЛЮЧЕНИЕ	43
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	44

Расширение функционала базы данных для встроенных приложений

АННОТАЦИЯ

Базы данных позволили снизить затраты, связанные с управлением данными, абстрагировав приложения от проблем, связанных с обработкой информации. Растёт потребность в управлении и анализе данных в небольших встроенных устройствах и сенсорных узлах. Из-за ограниченности ресурсов эти устройства обычно не имеют чётко определённых API для управления данными и стандартов, таких как реляционная модель и SQL. Это приводит к увеличению сложности и стоимости. LittleD - это реляционная база данных SQL, позволяющая выполнять специальные запросы на сенсорных устройствах. Новая реализация LittleD адаптируется к ограничениям памяти и размера кода путём оптимизации разбора и выполнения запросов и внедрения эффективных технологий управления памятью. Результаты экспериментов показывают, что LittleD выполняет запросы с объединениями и выборками на устройствах с памятью менее 2 КБ за несколько секунд.

Ключевые слова: сенсорный узел, микропроцессор, встроенное устройство, запрос, база данных.

ANNOTATION

Databases have reduced the costs associated with data management by abstracting applications away from the problems associated with processing information. There is a growing need for data management and analysis in small embedded devices and sensor nodes. Due to limited resources, these devices typically lack well-defined data management APIs and standards, such as the relational model and SQL. This leads to increased complexity and cost. LittleD is a relational SQL database that allows specific queries to be carried out on touch devices. The new implementation of LittleD adapts to memory and code size constraints by optimising

parsing and execution of queries and implementing efficient memory management techniques. Experimental results show that LittleD executes merge and fetch queries on devices with less than 2KB of memory in a few seconds.

Key words: sensor node, microprocessor, embedded device, query, database.

ВВЕДЕНИЕ

Реляционные базы данных и SQL позволили сократить затраты и время на управление и анализ данных. Всё большие объёмы данных собираются небольшими встроенными и сенсорными устройствами и находятся на серверах для анализа. Возможность выполнять анализ данных на устройстве снижает время задержки и передачи данных по сети, что делает устройства более энергоэффективными и надёжными. Кроме того, сбор данных может происходить без связи с сетью. Таким образом, адаптация технологии баз данных к данным маленьким устройствам является критически важной.

Несмотря на эту потребность, существует мало систем управления данными или API для встраиваемых и сенсорных устройств. Ресурс ограничения широко используемых сенсорных узлов и встроенных систем представляют собой серьёзную проблему. Самые ограниченные устройства могут иметь всего 16 КБ ПЗУ для скомпилированного кода, 2КБ ОЗУ и менее 1МВ постоянной памяти. Эти ограничения ресурсов сбавляют подходы к управлению данными и исключают использование распространённых встроенных баз данных, таких как SQLite, которые работают на более мощном оборудовании, например на смартфонах. Системы и алгоритмы также должны адаптироваться к характеристикам флэш-памяти. Флэш-память имеет асимметричную производительность: запись занимает на порядок больше времени, чем чтение.

Преыдушие системы, такие как PicoDBMS[2] и TinyDB[9], реализуют только механизм выполнения запросов на устройстве и полагаются на внешние данные для анализа, перевода и оптимизации плана запроса для выполнения на устройстве. Единственной системой, имеющей API, близкий к SQL, является Antelope[10], которая использует похожий, но не совместимый с SQL язык для создания специальных запросов и их выполнения. Antelope разработана для работы с Contiki[4] и имеет некоторые ограничения на обработку запросов.

Вклад этой работы заключается в создании SQL-совместимого механизма

запросов, который работает на устройствах с памятью всего 2 КБ. Отличительными особенностями LittleD являются:

- Новый рукописный синтаксический анализатор, минимизирующий потребление памяти.
- Система трансляции запросов, которая строит план выполнения запроса непосредственно во время разбора без необходимости строить деревья разбора и логические деревья запросов.
- Система гарантированного, фиксированного распределения памяти, которая обеспечивает выполнение запросов с заданным объёмом памяти при сохранении производительности.
- Возможность выполнять объединение двух или более таблиц, по возможности используя предикаты фильтрации и индексы.
- Поддержка оценки общих выражений.

LittleD был экспериментально оценён путём моделирования с помощью MSPSim на устройстве Zolertia Z1 и было сделано сравнение с Antelope[10]. Результаты экспериментов показывают, что LittleD постоянно использует меньше памяти и более гибок в обработке запросов, с более высокой степенью соответствия SQL. LittleD достигает эффективности использования ресурсов, значительно превосходящей эффективность Antelope.

1. Конструкторский раздел

1.1 Постановка задачи

Реализация LittleD имеет в себе довольно обширный функционал:

- CREATE — создание таблицы
- SELECT — поиск нужного элемента в таблице
- INSERT — вставка нового элемента в конец таблицы
- JOIN — найти и объединить таблицы по определённому условию
- SORT — отсортировать базу данных по определённому условию

Но для работы с ограниченной памятью необходимо иметь возможность удалять старые данные и вставлять на их место новые значения. Именно для этого я поставил задачу реализовать два новых функционала:

- UPDATE — обновить элемент таблицы
- DELETE — пометить на удаление элемент таблицы

1.2 RTEMS

Основой для проведения тестирования базы данных будет являться операционная система реального времени RTEMS.

Real-Time Executive for Multiprocessor Systems или RTEMS - это операционная система реального времени (RTOS) с открытым исходным кодом, которая поддерживает открытые стандартные интерфейсы прикладного программирования (API), такие как POSIX. Она используется в космических полётах, медицине, сетевых и многих других встраиваемых устройствах. В настоящее время RTEMS поддерживает 18 процессорных архитектур и около 200 BSP. Среди них ARM, PowerPC, Intel, SPARC, RISC-V, MIPS и другие. RTEMS включает несколько файловых систем, симметричную многопроцессорную обработку (SMP), встроенную оболочку, динамическую загрузку, а также высокопроизводительный, полнофункциональный стек IPV4/IPV6 TCP/IP из FreeBSD, который также обеспечивает RTEMS USB.

1.3 Real-time operating system (RTOS)

Операционная система реального времени (RTOS) - это ОС, которая

гарантирует приложениям реального времени определённые возможности в заданный срок. RTOS разрабатываются для критически важных систем и для устройств, таких как микроконтроллеры, которые зависят от времени. Требования к времени обработки RTOS измеряются в миллисекундах. Любая задержка с ответом может иметь катастрофические последствия.

Операционные системы реального времени имеют схожие функции с ОС общего назначения (GPOS), такими как Linux, Microsoft Windows или macOS, но разработаны таким образом, чтобы планировщик в ОС мог выполнять различные задачи в определённые сроки.

RTOS также часто используются во встроенных системах, которые представляют собой комбинацию аппаратного и программного обеспечения, предназначенного для выполнения определенной функции, а также могут работать в составе более крупной системы. Часто встроенные системы используются в средах реального времени и используют операционную систему реального времени для связи с оборудованием.

RTOS предназначены для одновременной обработки нескольких процессов, гарантируя, что эти процессы реагируют на события в течение предсказуемого времени. Обработка в RTOS происходит в рамках определенных временных ограничений и контролирует приоритет задач. RTOS также может вносить изменения в приоритет задач. Системы, управляемые событиями, часто переключаются между задачами на основе приоритета.

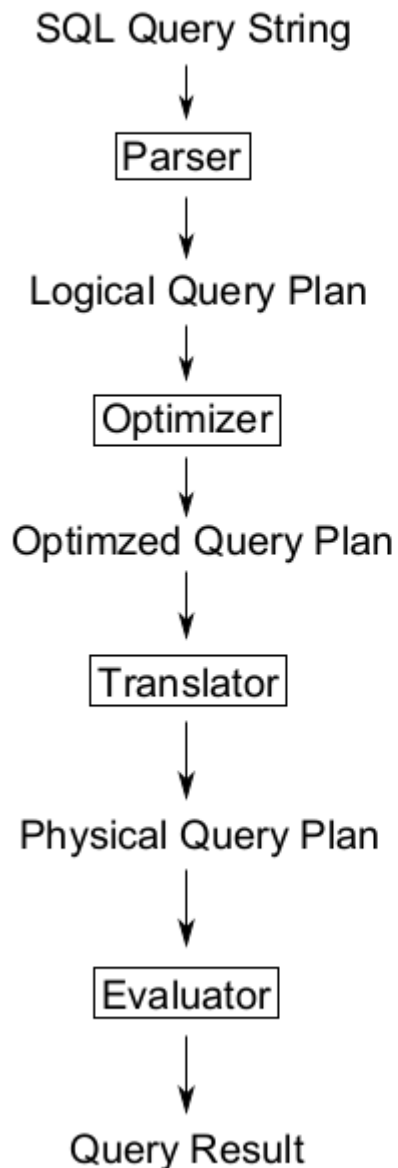
2. История

2.1 Описание основ

Базы данных хранят, извлекают и преобразуют данные. Реляционные системы управления базами данных (СУБД) обеспечивают свойства атомарности, согласованности, изоляции и долговечности (ACID), а также язык структурированных запросов (SQL) для определения схемы и данных, их поиска и преобразования. В идеале такие средства должны быть доступны и для реляционной СУБД для микропроцессоров и сенсорных узлов.

Реляционные базы данных, использующие SQL, переводят запрос, заданный пользователем, в исполняемую последовательность шагов (см. Рисунок 1). Синтаксический анализатор лексем строк запроса строит дерево разбора, а затем преобразует это дерево разбора в логический план запроса. Оптимизатор преобразует логический план запроса в оптимизированный план запроса, который затем переводится в физический (или исполняемый) план запроса (см. Рисунок 2). Операторы являются внутренними узлами дерева, а сканирование отношений – листьями. Затем оценщик выполняет программу для получения результатов.

Typical SQL Database



LittleD

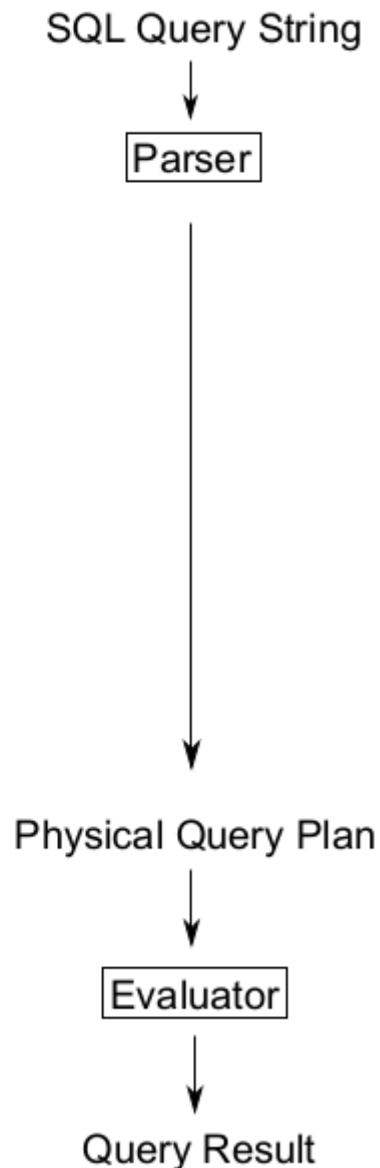


Рисунок 1: Архитектура процессора запросов для типичной базы данных SQL и LittleD

`SELECT R.x, T.y, T.a+T.b FROM R, T WHERE R.x=T.y`

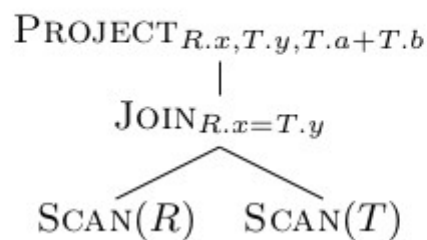


Рисунок 2: Пример исполняемого плана запроса.

В средах с ограниченными ресурсами такой процесс выполнения запроса нецелесообразен из-за использования памяти и ограничений на размер кода. Встроенные устройства могут иметь только от 16 до 128 КБ места для кода, и сложные компоненты, такие как оптимизатор, не могут быть реализованы. Кроме того, устройства могут иметь всего 2 КБ оперативной памяти. Генерация промежуточных репрезентаций, таких как логические и оптимизированные планы запросов, занимают слишком много места. Следовательно, системы вынуждены идти на компромиссы в отношении внедрённых компонентов и уровня поддержки SQL.

Одним из наиболее распространённых компромиссов является выполнение только выполнения запросов на устройстве. В этом случае запрос задаётся с помощью SQL-подобного языка на рабочей станции, а затем оптимизируется и переводится в исполняемый план на рабочей станции. Исполняемый план передаётся на встроенное устройство для выполнения. Устройство для выполнения. Хотя это устраняет значительную сложность кода за счёт удаления синтаксического анализатора и оптимизатора, встроенное устройство теперь становится зависимым от внешнего устройства. Это также добавляет ещё один уровень сложности при реализации алгоритмов работы с данными.

Такие системы, как COUGAR[3] и TinyDB[9], являются распределёнными системами данных, предназначенными для управления информацией о множества датчиков, объединённых в сеть. Эти системы выполняют разбор и трансляцию запросов вне устройства. Существует ведущая система, которая управляет запросами по всей сети.

Другая техника заключается в упрощении выполнения базы данных используемые алгоритмы и повторные вычисления, насколько это возможно. PicoDBMS[2] предоставляет основу для разработки алгоритмов выполнения запросов с меньшим объемом оперативной памяти. алгоритмов выполнения запросов с меньшим объемом оперативной памяти. Авторы демонстрируют что

ограничение алгоритмов абсолютным минимумом оперативной памяти, даже при наличии доступных индексов, приводит к значительным повторным вычислениям. Однако увеличение объема оперативной памяти, предоставляемой этим алгоритмам даже в скромных объемах, резко значительно повышает производительность алгоритмов. PicoDBMS разработана для таких устройств, как смарт-карты, которые имеют EEPROM и около 1 КБ памяти.

Для более крупных мобильных устройств, таких как смартфоны, существуют встроенные продукты баз данных. SQLite является популярной встроенной базой данных. Другие подобные СУБД SQL включают H2 Database Engine, SQL Server Compact и Mini SQL. Ни одна из этих систем не нацелена на самые ограниченные устройств, а нацелены на более мощные платформы, такие как мобильные телефоны. К не реляционным альтернативам относятся BerkeleyDB и UnQLite. Например, SQLite требует не менее 200 КБ пространства кода и не может работать на популярных платформах, таких как например, Arduino.

Для сенсорных устройств наиболее подходящей SQL-подобной базой данных является Antelope[10]. Antelope предоставляет синтаксический анализатор запросов и систему выполнения, которая может работать на небольших встроенных устройствах. устройствах и язык запросов под названием AQL, который похож на SQL. AQL разбивает утверждения на более мелкие фрагменты, чтобы минимизировать использования памяти. Запросы на получение данных пишутся в более явном виде. Например, для объединений выделяется своя команда. Хотя это позволяет делать более короткие запросы, это также менее гибким и не соответствует стандарту SQL.

2.2 SQLite

Стоит обратить внимание на программу SQLite, уже представленную на рынке. Это компактная встроенная СУБД, которая используется в браузерах, музыкальных плеерах и других программах, но для задачи установки на

микроконтроллер с ограниченным объемом флэш-памяти она не подходит. Тестовые испытания показали, что SQLite занимает 50 Кбайт постоянной памяти, а LittleD - 5 Кбайт.

ЗАКЛЮЧЕНИЕ


В данной работе представлена LittleD, реляционная база данных SQL для самых маленьких встроенных устройств. По функциональности и производительности LittleD не уступает Antelope, но при этом является более гибкой и использует значительно меньше памяти для обработки запросов. Будущие исследования будут сосредоточены на индексировании и алгоритмических улучшениях реляционных операторов для снижения потребления оперативной памяти и энергии. Это будет включать реализацию MinSort[5] для значительного повышения скорости сортировки без индексации. Различные методы разбора, от компиляции запросов вне устройства до новых представлений для SQL-подобных языков, будут исследованы для уменьшения требований к кодовому пространству полной системы. Будет проведена дальнейшая работа по использованию этой системы на распространенных платформах, таких как Arduino.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] D. Agrawal, D. Ganesan, R. Sitaraman, Y. Diao, and S. Singh. Lazy-adaptive tree: An optimized index structure for flash devices. Proc. VLDB Endow., 2(1):361–372, Aug. 2009.
- [2] N. Anciaux, L. Bouganim, and P. Pucheral. Memory Requirements for Query Execution in Highly Constrained Devices. VLDB '03, pages 694–705. VLDB Endowment, 2003.
- [3] P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. MDM '01, pages 3–14, London, UK, UK, 2001. Springer-Verlag.
- [4] Contiki Operating System. <http://www.contiki-os.org/>. Accessed: 2013-08-15.
- [5] T. Cossentine and R. Lawrence. Fast Sorting on Flash Memory Sensor Nodes. IDEAS '10, pages 105–113, New York, NY, USA, 2010. ACM.
- [6] E. W. Dijkstra. Algol 60 translation : An Algol 60 translator for the x1 and Making a translator for Algol 60. Technical Report 35, Mathematisch Centrum, Amsterdam, 1961.
- [7] E. Gal and S. Toledo. Algorithms and Data Structures for Flash Memories. ACM Comput. Surv., 37(2):138–163, June 2005.
- [8] D. Kang, D. Jung, J.-U. Kang, and J.-S. Kim. M-tree: An Ordered Index Structure for NAND Flash Memory. EMSOFT '07, pages 144–153, New York, NY, USA, 2007. ACM.
- [9] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. ACM Trans. Database Syst., 30(1):122–173, Mar. 2005.
- [10] N. Tsiftes and A. Dunkels. A Database in Every Sensor. SenSys '11, pages 316–332, New York, NY, USA, 2011. ACM.
- [11] Burns, Alan & Wellings, Andrew: Real-Time Systems and Programming Languages (3rd ed), Addison Wesley, 2001
- [12] Kopetz, Hermann: Real-time Systems : Design Principles for Distributed Embedded Applications, Kluwer Academic Publishers, 1997
- [13] Joseph, Mathai: Real-time Systems: Specification, Verification and Analysis, 2001
- [14] Nicolas Anciaux: PicoDBMS: Validation and Experience

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

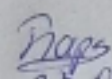
УТВЕРЖДАЮ
Заведующий кафедрой
 В.В. Шайдуров
(подпись) инициалы фамилия
«23» июня 2023 г.

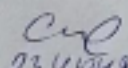
МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

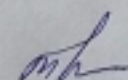
Расширение функционала базы данных для встроенных приложений

Направление 02.04.01 Математика и компьютерные науки

Магистерская программа 02.04.01.01 Математическое и компьютерное
моделирование

Руководитель  23.06.2023 доцент, кандидат физико-
математических наук С.Н. Баранов

Выпускник  23 июня 2023 г. И.Б. Смирнов

Нормоконтролер  23.06.2023 г. Т.Н. Шипина

Красноярск 2023