УДК 004.62; 004.91

И. О. Сычёв, С. А. Кондратьев, Ю. А. Кораблёв
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Методы управления логированием в информационно-управляющих системах на основе встраиваемых баз данных

Рассматриваются вопросы управления документированием данных в информационно-управляющих системах. Актуальность вопросов документирования связана с необходимостью хранения и обработки больших объемов данных в системах различных классов, в частности в системах автоматизированного контроля и диагностики. Определены общие свойства и требования к подсистеме документирования. Проведен анализ методов хранения регистрационных журналов и инструментов для сбора данных и их обработки. Рассмотрены преимущества и недостатки хранения и документирования данных в реляционных базах данных. Предложен подход к хранению информации с использованием встраиваемых баз данных «ключ–значение». Рассмотрены примеры реализации на примере СУБД libmdbx и SQLite. Разработана и представлена структура хранилища регистрационных журналов. Разработан программный модуль документирования информации, представлена архитектура модуля. Рассмотрен модуль документирования как часть программного комплекса хранения и анализа данных.

Документирование, логирование, база данных, диагностика, NoSQL

Процесс управления регистрацией событий или документированием данных является важной частью современных информационных систем (ИС). Документирование рассматривается как общий механизм управления документированием информации в сложной технической системе. Процесс управления документированием определяет набор процессов и политик, используемых для администрирования, генерации, передачи, анализа, хранения, архивирования и конечного размещения больших объемов документируемых данных в информационной системе [1]. Регистрационный журнал (лог) – хронологически упорядоченная совокупность записей результатов деятельности субъектов системы, достаточная для восстановления, просмотра и анализа последовательности действий [2].

Процесс ведения регистрационного журнала включает следующие этапы:

– сбор и хранение;
– защита;
– интеграция;
– анализ.

Регистрационные журналы хранятся и обрабатываются в системах различного назначения, например в вычислительных сетях Internet of things (IOT), обеспечивающих взаимодействие компьютеров, объектов и людей без необходимости взаимодействия «человек–человек» или «человек–машина» [2], и SCADA – категории программного обеспечения, предназначенного для сбора данных в реальном времени с удаленных местоположений, чтобы контролировать оборудование и условия.

Основные особенности хранения логов. В любой сложной технической системе важнейший элемент – программный модуль (ПМ) документирования, поскольку только документирование позволяет определить *что*, *когда* и *как* произошло в системе. Документирование используется для анализа действий операторов; обнаружения и анализа ошибок; для проверки работоспособности системы и, в случае отказа в системе, для поиска причины отказа. Сам по себе сервис документирования может поставлять данные на вход другим программным модулям, таким, как ПМ анализа данных или ПМ диагностики; т. е. данные документирования служат основой построения сложных аналитических и диагностических систем. Это особенно важно в контексте использования данных для диагностики структурно-сложных систем, т. е. систем, которые при математическом описании не сводятся к последовательным, параллельным или древовидным структурам [3].

Процесс документирования должен потреблять минимальное количество ресурсов, не оказывая существенного влияния на другие функции системы и должен быть простым для использования и поддержки: это включает в себя простоту конфигурации, поддержку разных типов хранимых данных и простой программный интерфейс (API).

Как итог выделим следующие основные свойства ПМ документирования:

1. *Легкость в поддержке и конфигурации*. БД можно установить на любое устройство ИС и подключать к ней модули, которые должны записывать данные максимально прозрачно для клиентов.

2. *Запись данных с временной меткой*. Временная метка – основное поле для поиска в БД.

3. *Поддержка хранения разных форматов входной информации* (JSON, XML, бинарные данные), обеспечение функций для обработки хранимой информации.

4. *Основные DML операции – вставка (insert) и просмотр (select)*. Логи обычно удаляются не выборочно, а большими пачками. Обновление данных (update) в рассматриваемом случае не выполняется.

Формат хранимых данных. Формат данных документирования обычно определяется требованиями к ИС и протоколами обмена, используемыми в системе. Данные могут передаваться в текстовом (XML, JSON) или бинарном виде. Преимущества текстовых форматов обмена – легкость для обработки и чтения данных, готовые инструменты для разбора. Основные недостатки: избыточность – большой объем данных, занимаемый на диске; малая скорость, поскольку требуется преобразовывать данные в текстовый вид.

Бинарные протоколы эффективнее с точки зрения скорости и экономии памяти и ресурсов, но сложнее для поддержки и обработки: бинарные данные, содержащиеся в хранилище, можно прочитать только с помощью специальных программ или библиотек для чтения протокола. При документировании бинарных данных появляется еще одна проблема: изменение формата протокола приводит и к изменению библиотеки разбора пакетов, из-за чего необходимо поддерживать также и набор библиотек для разных версий протокола. Библиотеку разбора пакетов целесообразно реализовать как динамическую или как плагин – это позволит выбирать нужную версию библиотеки в зависимости от номера протокола.

Формат хранения данных определяется архитектурными требованиями к системе: ПМ документирования должен обеспечить возможность документирования разных типов данных.

Технические особенности хранения логов в реляционных базах данных. Общая концепция реляционных баз данных подходит для систем со сложноструктурированной информацией, где требуется полная поддержка DML (Data manipulation language – язык манипулирования данными): выборка, вставка, обновление и удаление информации. Высокая гибкость работы с данными и надежность реляционных БД достигаются за счет:

– журнала предзаписи WAL (write-ahead-log), обеспечивающего безопасность хранения данных;

– схемы базы данных, включающей в себя описание содержания, структуры и ограничения целостности;

– использования SQL-запросов для выполнения DML.

Идея WAL заключается в том, что изменения в файлах данных должны записываться только после того как изменения попали в журнал. Правило опережающего протоколирования (write-ahead logging rule) формулируется следующим образом:

Прежде чем транзакция T сможет модифицировать элемент X базы данных на диске, необходимо внести в копию протокола на диске все записи, имеющие отношение к операции модификации X, включая запись обновления вида <*T*, *X*, *v*> и запись фиксации <COMMIT T>.

Благодаря WAL нет необходимости записывать страницы данных на диск после каждой транзакции. Если произойдет сбой, данные можно будет восстановить, используя журнал. За счет WAL отсутствует необходимость использования журналируемой файловой системы (ФС) и присутствует возможность непрерывного архивирования и восстановления.

Формат входной информации для подсистемы логирования зависит от формата обмена, используемого в системе. Если преобразовывать входную информацию в реляционную модель, то схема данных будет меняться каждый раз, когда изменяется формат передаваемых данных, т. е. при этом облегчается выборка данных за счет возможностей языка SQL, но усложняется процесс записи данных и теряется гибкость при конфигурировании подсистемы.

Большинство реляционных БД (например, PostgreSql) – клиент-серверные. Выделенный сервер СУБД ограничивает возможность распределенного хранения информации в системе. Сервер БД необходимо конфигурировать для достижения максимальной производительности, обеспечивать сборку мусора и т. п., что обычно требует поддержки опытного администратора БД. Сборка мусора обусловлена тем, что при обычных операциях кортежи, удаленные или устаревшие в результате обновления, физически не удаляются из таблицы, а сохраняются в ней, пока не будет выполнена команда VACUUM [5]. Таким образом, периодически необходимо выполнять VACUUM, особенно для часто изменяемых таблиц.

РБД эффективны в системах со сложной бизнес-логикой, где требуется обрабатывать транзакции и необходимы возможности языка SQL. Запись логов в РБД сопряжена со следующими трудностями:

– возможное снижение производительности в связи с обработкой и планированием SQL-запросов;

– дополнительная нагрузка на диск из-за использования WAL;

– отсутствие необходимости использования SQL для выборки данных, особенно сложных конструкций с соединением таблиц.

Хранение логов в хранилище «ключ–значение». БД «ключ–значение», или хранилище «ключ–значение», – это парадигма хранения данных, разработанная для хранения ассоциативных массивов (хэш-таблиц или словарей), извлечения и управления ими [6]. Словари содержат коллекцию объектов; каждый объект имеет набор полей, содержащих данные. Хранилища «ключ–значение» используют доступ по ключу, обычно имеют высокую производительность и легко масштабируются [7].

Значением может быть двоичный объект, текст, документ в формате JSON или XML. Далее рассмотрен вариант реализации на примере встраиваемой БД libmdbx. Libmdbx – компактная, встраиваемая транзакционная БД «ключ–значение», сфокусированная на создании легковесных решений с высокой производительностью. Встраиваемые БД поставляются в виде библиотеки и не имеют выделенного сервера.

Отличительная особенность libmdbx – отсутствие журнала предзаписи (WAL), который является неотъемлемой частью реляционных БД. Как и ее предшественник LMDB, libmdbx использует отображение файлов в память – механизм работы с файлами, при котором всему файлу или некоторой непрерывной его части ставится в соответствие определенный участок памяти. Это значит, что libmdbx возвращает указатели на адреса ключей и значений, избегая ненужной и дорогой процедуры копирования памяти. Так же, как РБД, libmdbx поддерживает ACID-требования к транзакциям: атомарность, согласованность, изолированность, долговечность.

БД обеспечивает высокую гибкость, но обладает рядом ограничений. Отсутствие WAL приводит к увеличению коэффициента усиления записи (WAF – Write Amplification Factor). Высокий WAF означает, что запись данных на диск может быть ресурсоемкой.

Коэффициент усиления записи рассчитывается по формуле

$$\mathrm{WAF} = \frac{\text{данные, записанные на диск}}{\text{данные, записанные хостом}}.$$

К проблемам этой БД относится поддержка длинной транзакции на чтение, например горячее архивирование или отладка клиентского приложения: старые страницы не могут быть восстановлены, пока старая транзакция их использует.

В libmdbx применяются следующие методы: механизм копирования при записи COW (Copy-On-Write) – при чтении области данных используется общая копия, в случае изменения данных создается новая копия; MVCC (multi version concurrency control) – управление параллельным доступом с помощью многоверсионности: каждому пользователю предоставляется «снимок БД», вносимые изменения не видны другим пользователям до момента фиксации транзакции.

Реализация хранилища логов. Схема ПМ документирования представлена на рис. 1. Программное обеспечение разработано на языке C++ под операционную систему Linux. Рассматривается вариант хранения бинарных данных, но предлагаемая логика может применяться и для других типов данных.

Основная часть платформы документирования – компонент чтения и записи данных. Модуль записи логов обеспечивает совместный сетевой доступ нескольких приложений. Потребители информации – читатели – обращаются к БД с помощью клиентской библиотеки, которая взаимодействует с модулем чтения логов по протоколу TCP. Модули

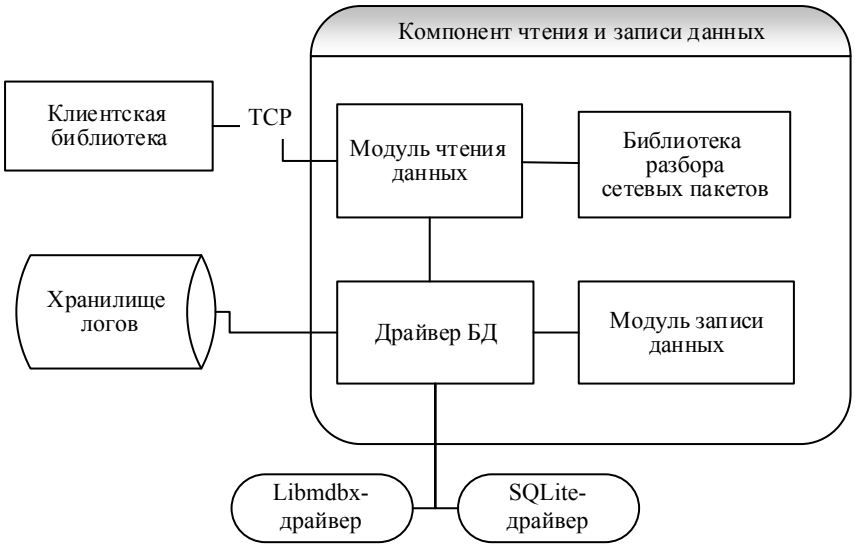

Рис. 1

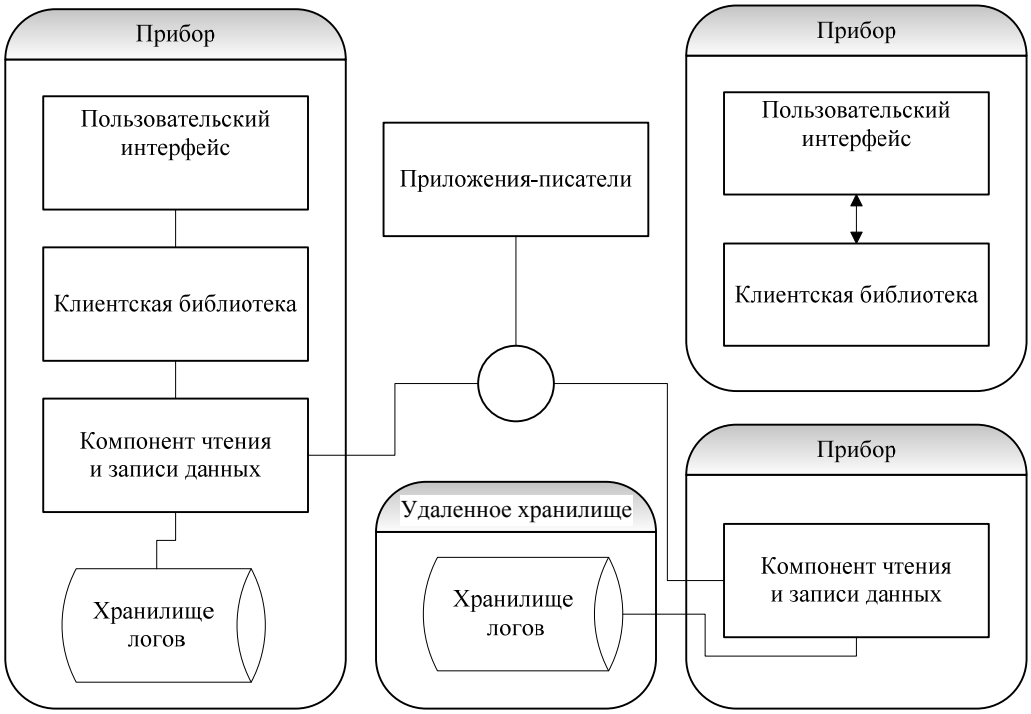

Рис. 2

могут быть расположены на нескольких приборах, и каждый из них может вести запись и чтение информации независимо.

Из бинарных пакетов информация извлекается с помощью динамической библиотеки для разбора сетевых пакетов. Библиотека используется на клиенте для разбора данных, полученных от сервера, и на сервере для выполнения фильтрации.

На рис. 2 представлены примеры вариантов конфигурации подсистемы:

1) получение данных от писателей и локальный доступ к информации для чтения;

2) получение данных от писателей и запись данных в хранилище, расположенное на удаленном диске; удаленный доступ к информации читателей.

Структура, созданная для хранения бинарных логов, описана в табл. 1.

Предложенная схема хранения данных обеспечивает следующие возможности для интерфейса:

– переход по времени с использованием индекса;

– переход по номеру пакета с использованием индекса;

– определение количества записей за заданный интервал времени с использованием индекса.

Важно, что предлагаемая архитектура может применяться с разными типами БД, используе-

Таблица 1

Таблица	Ключ	Значение
Таблица с данными	Идентификатор записи – беззнаковое целое	Информационный бинарный массив с пакетом данных
Индексная таблица времени	Время	Идентификатор записи в таблице с данными, соответствующий заданному времени
Индексная таблица типов	Тип данных	Массив идентификаторов записей в таблице с данными, содержащих данный тип
Таблица номеров записей	Тип данных	Массив упакованных в одно 64-битное число порядкового номера записи и идентификатора записи в таблице с данными

Таблица 2

Наименование таблицы БД	Название поля	Тип поля	Описание
DATA	DATAID	INT	Идентификатор пакета
	DATA	TINYBLOB	Пакет данных
TYPE	TYPE	INT	Тип данных
	DATA_DATAID	TINYBLOB	Массив пар «идентификатор и время регистрации» для пакетов типа TYPE фиксированного размера
	N	INT	Порядковый номер первого пакета в массиве DATA_DATAID
	I	INT	Идентификатор первого пакета в массиве DATA_DATAID
TIME	TIME	DATETIME	Дата/время
	DATA_DATAID	INT	Идентификатор пакета, последнего на момент времени TIME

мыми в качестве хранилища данных. Например, описанная конфигурация может быть реализована в SQLite. Таблицы в SQLite аналогичны таблицам libmdbx. ER-диаграмма хранилища показана на рис. 3.

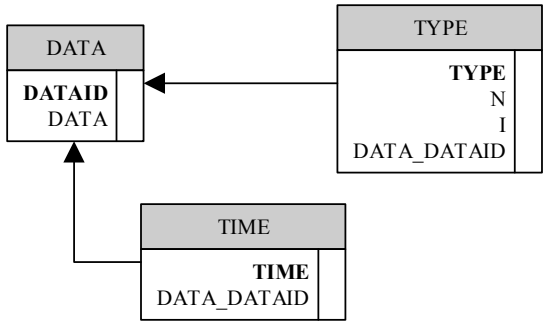

Рис. 3

Описание полей структуры данных SQLite представлено в табл. 2.

Обращение логов к БД осуществляется через драйвер БД. В случае изменения типа хранилища данных в ПМ документирования необходимо изменить драйвер БД.

Внутреннее устройство. В целях уменьшения объема данных, потерянных при повреждении накопителя, использования широко распространенных Flash-носителей небольшого объема для переноса данных и упрощения процедуры резервного копирования выбрана схема многотомного документирования. Лог разбивается на файлы-тома размером по 500 Мбайт, новый том создается автоматически при заполнении старого.

Для программы-клиента переход между томами прозрачен, поскольку сервер обеспечивает сквозную нумерацию строк данных в рамках заданного оператором временного интервала. Для этого применен сортированный ассоциативный массив (контейнер `std::map` в STL), где ключом служит максимальный номер записи в сквозной нумерации по интервалу, а значением – пара <номер тома, смещение относительно начала тома>. Далее представлен алгоритм формирования ассоциативного массива для поиска записи.

Исходные параметры: начало интервала, конец интервала, количество записей в томе, тип данных.

Результат: ассоциативный массив T

$n = 0$;
цикл *i = ПервыйТомВИнтервале* до *ПоследнийТомВИнтервале* **выполнять**
  Установить *i* том;
  если в томе есть данные запрошенного типа **тогда**
    если *i = ПервыйТомВИнтервале* и *i = ПоследнийТомВИнтервале* **тогда**
      l = ПоследняяЗаписьИнтервала - ПерваяЗаписьИнтервала + 1;
    иначе если *i = ПервыйТомВИнтервале*
      l = КоличествоЗаписейДанногоТипаВТоме – ПерваяЗаписьИнтервала + 1;

```
    иначе если i = ПоследнийТомВИнтервале
        l = ПоследняяЗаписьИнтервала;
    иначе
        l = КоличествоЗаписейДанногоТипаВТоме;
    конец
    k = n + l – 1;
    если i = ПервыйТомВИнтервале
        v = <i, 1 - ПерваяЗаписьИнтервала>;
    иначе
        v = <i, n>;
    конец
    Записать <k, v> в T;
    n = n + l;
конец
```

Используя данный массив, можно узнать для произвольной строки заданного интервала номер тома и номер записи. Для этого определяется значение по первому ключу массива, большему либо равному номеру записи в сквозной нумерации. Тогда номер записи относительно начала тома $i = n – o$, где n – номер в сквозной нумерации, o – смещение.

Вычислительная сложность определения тома и номера записи в нем задается сложностью алгоритма `std::lower_bound` по контейнеру `std::map` STL и составляет $O(\log(n))$.

Результаты. В результате выполненного исследования был реализован ПМ документирования для бинарных данных с поддержкой двух хранилищ данных: mdbx и SQLite. Эксперимент проводился на вычислительной системе, оборудованной 2-ядерным процессором Intel Core 2 Duo CPU E6550 и 8 Гбайт оперативной памяти, операционная система Linux Debian. Результаты измерения производительности:

– mdbx – 14 Мбайт ОЗУ; 100 Мбайт/ч заполнение файла; скорость записи ~2 Мбайт в 30 с;

– sqlite3 – 7 Мбайт ОЗУ; 60 Мбайт/ч заполнение файла; скорость записи ~2 Мбайт в 30 с.

Видно, что присутствует незначительная разница: SQLite потребляет меньше ОЗУ и дискового пространства.

За счет модульного построения платформа может быть сконфигурирована под требования в конкретной системе. При этом определяются способы обмена данными в системе; формат записи и хранения данных (бинарный формат, JSON, XML); тип хранилища данных (libmdbx или SQLite). Вне зависимости от типа хранилища информации данные хранятся с помощью пар «ключ–значение». При использовании SQLite применяются только механизмы, обеспечивающие безопасность и надежность хранения данных, а не все функциональные возможности СУБД; логика и обработка схемы данных реализованы в модулях записи и чтения логов и библиотеке для разбора сетевых пакетов.

Предложенные подходы пригодны для документирования информации в сложных технических системах различного назначения, в том числе в интеллектуальных системах отказоустойчивого управления [8] и интеллектуальных географических информационных системах (ГИС). В ГИС ПМ документирования может быть использован в качестве инструмента сбора, обработки и хранения информации от всех источников информации и взаимодействующих систем [9]. ПМ документирования рассматривается прежде всего как часть программного комплекса (ПК) хранения и анализа данных, схема которого представлена на рис. 4.

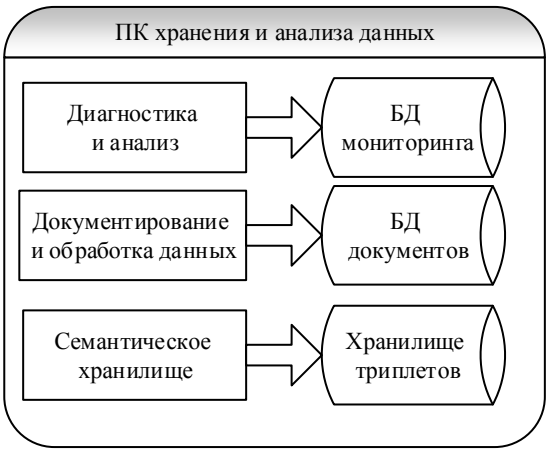

Рис. 4

ПК включает в себя следующие модули: диагностика и анализ; документирование и обработка данных; семантическое хранилище. Модуль семантического хранилища реализован [10] в модели триплетов RDF (Resource description framework – среда описания ресурса). Модуль обеспечивает семантику, необходимую для непротиворечивой машинной обработки данных и знаний о предметной области, а также развитые возможности извлечения сложных закономерностей в данных [11].

СПИСОК ЛИТЕРАТУРЫ

1. Управление логами. URL: https://searchitoperations.techtarget.com/definition/log-management (дата обращения 20.12.2019).

2. Гладких А. А., Дементьев В. Е. Базовые принципы информационной безопасности вычислительных сетей: учеб. пособие / УлГТУ. Ульяновск, 2009.

3. Рябинин И. А. Структурно-сложные системы и их формализация с помощью функций алгебры логики // Биосфера. 2011. Т. 3, № 4. С. 455–461.

4. Гарсиа-Молина Г., Ульман Дж. Д., Уидом Дж. Системы баз данных. Полный курс. М.: Издательский дом «Вильямс», 2003. С. 839–841.

5. Документация PostgeSql. URL: https://www.postgresql.org/docs/12/index.html (дата обращения 20.12.2019).

6. База данных «ключ–значение». URL: https://en.wikipedia.org/wiki/Key-value_database (дата обращения 20.12.2019).

7. Садаладж П. Дж., Фаулер М. NoSQL Новая методология разработки нереляционных баз данных. М.: Издательский дом «Вильямс», 2013. С. 101–102.

8. Кораблев Ю. А. Проектирование гибридных интеллектуальных систем отказоустойчивого управления. СПб., 2019. С. 3.

9. Попович В. В. Интеллектуальные географические информационные системы. СПб.: Наука, 2013. С. 37–50.

10. Кондратьев С. А., Сычев И. О. Применение семантических технологий для хранения и обработки данных в корабельных информационно-управляющих системах // Морская радиоэлектроника. 2019. № 2 (68). С. 38–41.

11. Сычев И. О., Кораблев Ю. А., Звягин Л. С. Применение семантических технологий для обработки данных в геоинформационных системах // Всерос. науч. конф. по пробл. управления в технических системах. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2019. Т. 1. С. 322–325.

I. O. Sychev, S. A. Kondratev, Ju. A. Korablev
Saint Petersburg Electrotechnical University

LOGGING MANAGEMENT METHODS IN INFORMATION MANAGEMENT SYSTEMS BASED ON EMBEDDED DATABASES

The issues of information logging management in information management systems are considered. Relevance of logging issues is related to the need to store and process large amounts of data in systems of various classes, particularly in automated monitoring and diagnostics systems. The general requirements for loggings software module are defined. The analysis of methods for log storage and tools for their collecting and processing are carried out. The method for storing logs using «key-value» database is proposed. The implementation variants using DBMS limbmdbx and SQLite are considered. Storage structure for logs is developed. Logging software module is developed; module architecture is carried out.

Documenting, logging, data base, diagnostics, NoSQL

УДК 519.67

Д. В. Козлов, Я. А. Бекенёва
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Интеллектуальный анализ данных в области определения пригодности собак к выполнению специфической службы

Проведение различного рода тестирований широко используется в качестве инструмента оценки в различных областях деятельности. Результаты тестирований, как правило, хранятся в специальных базах и содержат информацию о тестируемых объектах, результатах для каждого отдельного теста и итоговый результат. Объем данных о результатах тестирований может быть достаточно большим и с трудом поддаваться обработке вручную. Объектом разработки и исследования служат математические модели классификации, применяемые к исходному набору данных с характеристиками собак, выполняющих определенный род деятельности. Целью статьи является исследование возможности применения методов классификации для оценки пригодности собак к выполнению службы собаки-поводыря. Были исследованы несколько методов классификации: k ближайших соседей, дерево классификации, логистическая регрессия. Разработана программа для подготовки исходного набора данных к дальнейшему интеллектуальному анализу с помощью выбранных методов. Проведены эксперименты, связанные с оценкой точности каждого метода, а также выявлены критерии, имеющие наибольшее влияние на результаты тестирования собак.

Классификация, метод *k* ближайших соседей, дерево классификации, логистическая регрессия

Одним из основных подходов к машинному обучению является обучение с учителем (supervised learning) [1]. Данный подход характеризуется наличием некоторой выборочной совокупно-