Оглавление

Введение

Рабочая программа дисциплины – программа освоения учебного материала, соответствующая требованиям государственного образовательного стандарта высшего профессионального образования и учитывающая специфику подготовки студентов по избранному направлению или специальности. Разрабатывается для каждой дисциплины учебного плана всех реализуемых в университете основных образовательных программ [1].

Хранение, обработка и анализ информации, находящейся в рабочей программы дисциплины может пригодиться для различных систем, например, системы управления обучения (англ. Learning Managment System (LMS) [2]). Такой интерфейс может предоставить пользователю системы (в данном случае преподавателю) получать и редактировать информацию о рабочей программы дисциплины в режиме онлайн, например, в личном кабинете пользователя.

Рабочая программа дисциплины обычно представлена в виде документа в формате Microsoft Word [3], что накладывает ограничения на автоматизированную программную обработку и анализ информации, предоставленной в рабочей программы дисциплины.

Цель работы – реализовать программное обеспечение для хранения, редактирования и удаления данных о рабочих программах дисциплин.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- проанализировать варианты представления данных и выбрать подходящий вариант для решения задачи;
- проанализировать системы управления базами данных и выбрать подходящую систему для хранения данных;
- спроектировать базу данных, описать ее сущности и связи;
- реализовать интерфейс для доступа к базе данных;
- реализовать программное обеспечение, которое позволит получить доступ к данным по средствам REST API [4].

1 Аналитическая часть

В данном разделе описана структура рабочей программы дисциплины. Представлен анализ способов хранения данных и систем управления базами данных, оптимальных для решения поставленной задачи. Описаны проблемы кэшированных данных и представлены методы их решения.

1.1 Формализация задачи

Каждая дисциплина, преподаваемая в высшем учебном заведении, имеет свою рабочую программу. В ней хранятся различная информация о дисциплине: стандарт, содержание, объем, результаты обучения, перечень литературы, методические указания и прочее. Зачастую, у пользователей нет никаких автоматизированных инструментов для анализа и редактирования таких программ.

Дисциплина имеет свой федеральный государственный образовательный стандарт: 3+, 3++ и другие [5]. Образовательный стандарт – это совокупность обязательных требований к образованию определенного уровня и (или) к профессии, специальности. Каждый образовательный стандарт для каждого направления подготовки обучаемого имеет свою компетенцию. Компетенция – некоторый свод информации, о том что должен знать, уметь и какими навыками должен обладать выпускник, успешно осовевший дисциплину.

Каждому направлению подготовки, но одной и той же рабочей программы дисциплины сопоставлены различные компетенции (которые, в свою очередь, различны в каждом образовательном стандарте). Например, есть два студента, успешно осовевшие дисциплину «Физика». Один из них обучается по направлению подготовки «Экономика», а второй по направлению «Теплофизика и теоретическая теплотехника». Очевидно, что второй студент должен владеть большими знаниями о данной дисциплине.

Коды компетенций для каждого образовательного стандарта отличаются. К сожалению, не редки случаи, когда при переходе на новую образовательную программу меняется только код и название компетенции – содержание компетенции остается абсолютно точно таким же. В связи с

этим, возникает возможность автоматизации перевода документов рабочей программы дисциплины на новый стандарт.

Кроме того, имея данные о всех рабочих программах дисциплинах, можно адаптировать какие-либо из них под выбранные направления подготовки. Например, рассчитать оптимальную нагрузку по данной дисциплине для студентов обучающихся на данном направлении.

Далее, в качестве примера, будем рассматривать рабочую программу дисциплины «Информатика», соответствующую образовательному стандарту 3++, разработанную и преподаваемую в МГТУ им. Н. Э. Баумана [6].

1.2 Структура рабочей программы дисциплины

Структура файлов рабочих программ дисциплин можно разниться от ВУЗа к ВУЗу, но, внутри одного ВУЗа, скорее всего, все программы имеют одну и ту же (или схожую) структуру. Рабочая программа дисциплины «Информатика» имеет следующие разделы:

1. титульный лист;
2. планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы;
3. место дисциплины в структуре образовательной программы;
4. объем дисциплины;
5. содержание дисциплины;
6. учебно-методическое обеспечение самостоятельно работы;
7. фонд оценочных средств для проведения текущего контроля и промежуточной аттестации студентов;
8. перечень основной и дополнительной литературы;

9. методические указания;

10. перечень информационных технологий;

11. описание материально-технической базы.

Разделы №2, №4, №5 представленны в виде совокупности текстовой информации и таблиц (рис. 1.1). Остальные разделы представлены в виде текстовой информации (рис. 1.2).

3. ОБЪЕМ ДИСЦИПЛИНЫ

Общий объем дисциплины составляет 7 зачетных единиц(з.е.), 252 академических часа.

В том числе:
1 семестр – 4 з.е. (144 ак.ч.),
2 семестр – 3 з.е. (108 ак.ч.).

Таблица 2. Объём дисциплины по видам учебных занятий (в академических часах)

Виды учебной работы	**Объем по семестрам, ч**		
	Всего	**Семестры**	
		1	**2**
Объем дисциплины	252	144	108
Аудиторная работа	**136**	**85**	**51**
Лекции (Л)	17	17	0
Семинары (С)	51	34	17
Лабораторные работы (ЛР)	68	34	34
Самостоятельная работа (СР)	**116**	**59**	**57**
Проработка учебного материала лекций	2	2	0
Подготовка к семинарам	6.25	4.25	2
Подготовка к лабораторным работам	50	16	34
Подготовка к экзамену	30	30	0
Подготовка к рубежному контролю	18	6	12
Другие виды самостоятельной работы	9.75	0.75	9
Вид промежуточной аттестации		**Экзамен**	**Зачёт**

Рис. 1.1: Изображение таблицы с информацией о объеме дисциплины.

Интерес представляют разделы №1 (титульный лист), №2 (результаты обучения), №4 (объем дисциплины), №5 (содержание дисциплины) и №8 (перечень литературы).

Первый раздел содержит общую информацию о курсе: название, образовательный стандарт и прочее.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина входит в блок Б1 «Дисциплины (модули)» образовательных программ бакалавриата по направлениям 11.03.04 «Электроника и наноэлектроника», 12.03.02 «Оптотехника», 12.03.05 «Лазерная техника и лазерные технологии», 13.03.02 «Электроэнергетика и электротехника», 13.03.03 «Энергетическое машиностроение», 14.03.01 «Ядерная энергетика и теплофизика», 24.03.01 «Ракетные комплексы и космонавтика», 28.03.02 «Наноинженерия».

Изучение дисциплины предполагает предварительное освоение следующих дисциплин (в рамках школьного курса):

- Основы информатики;
- Математика;
- Иностранный язык (английский язык).

Освоение данной дисциплины необходимо как предшествующее для следующих дисциплин образовательной программы:

- Инженерная и компьютерная графика

Освоение учебной дисциплины связано с формированием компетенций с учетом матриц компетенций ОПОПдля направлений (уровень бакалавриата): 11.03.04 «Электроника и наноэлектроника», 12.03.02 «Оптотехника», 12.03.05 «Лазерная техника и лазерные технологии», 13.03.02 «Электроэнергетика и электротехника», 13.03.03 «Энергетическое машиностроение», 14.03.01 «Ядерная энергетика и теплофизика», 24.03.01 «Ракетные комплексы и космонавтика», 28.03.02 «Наноинженерия».

Рис. 1.2: Изображение текстовой информации о месте дисциплине в структуре образовательной программы.

Раздел №2 содержит коды и описания компетенций для каждого направления подготовки. С помощью информации, полученной в этом разделе, можно попробовать автоматизировать перенос файла рабочей программы дисциплины с одной образовательной программы на другую.

В разделах №4 и №5 хранится информация о нагрузке и структуре дисциплины – эта информация может пригодится для анализа нагрузки на студентов по выбранному направлению подготовки и структуризации рассматриваемой рабочей программы дисциплины.

1.3 Базы данных и системы управления базами данных

В задаче разбора и хранения информации рабочей программы дисциплины важную роль имеет выбор модели хранения данных. Для персистентного хранения данных используются базы данных [7]. Для управления этими базами данных используется системы управления данных – СУБД [8]. Система управления базами данных – это совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

1.4 Хранение данных о рабочих программах дисциплины

Система, разрабатываемая в рамках курсового проекта, предполагает собой приложение, которое является микросервисом [9] одной большой системы – системы управления обучения.

Предполагается, что доступ к разрабатываемому приложению будем иметь лишь только «ядро» этой системы. При этом, только у одного типа пользователя системы есть доступ к данным, хранящимся в приложении – преподавателю. Состояние гонки (англ. Race condition [10]) можно исключить - каждый преподаватель работает только с информацией из файлов, которые он самостоятельно загрузил в базу данных.

Для хранения данных о рабочей программы дисциплины необходимо использовать строго структурированную и типизированную базу данных, потому что вся информация, предоставленная в файлах программы имеет чётко выраженную структуру, которая не будет меняться от дисциплины к дисциплине.

1.4.1 Классификация баз данных по способу хранения

Базы данных, по способу хранения, делятся на две группы – строковые и колоночные. Каждый из этих типов служит для выполнения для определенного рода задач.

Строковые базы данных

Строковыми базами даных называются такие базы данных, записи которых в памяти представляются построчно. Строковые баз данных используются в транзакционных системах (англ. OLTP [11]). Для таких систем характерно большое количество коротких транзакций с операциями вставки, обновления и удаления данных - `INSERT`, `UPDATE`, `DELETE`.

Основной упор в системах OLTP делается на очень быструю обработку запросов, поддержание целостности данных в средах с множественным доступом и эффективность, которая измеряется количеством транзакций в секунду.

Схемой, используемой для хранения транзакционных баз данных, является модель сущностей, которая включает в себя запросы, обращающиеся к отдельным записям. Так же, в OLTP-системах есть подробные и текущие данных.

Колоночные базы данных

Колоночными базами данных называются базы данных, записи которых в памяти представляются по столбцам. Колоночные базы данных используется в аналитических системах (англ. OLAP [12]). OLAP характеризуется низким объемом транзакций, а запросы часто сложны и включают в себя агрегацию. Время отклика для таких систем является мерой эффективности.

OLAP-системы широко используются методами интеллектуального анализа данных. В таких базах есть агрегированные, исторические данные, хранящиеся в многомерных схемах.

1.4.2 Выбор модели хранения данных для решения задачи

Для решения задачи построчное хранение данных преобладает над колоночным хранением по нескольким причинам:

- задача предполагает постоянное добавление и изменение данных;
- задача предполагает быструю отзывчивость на запросы пользователя;
- задача не предполагает выполнения аналитических запросов;

1.4.3 Обзор СУБД с построчным хранение

В данном подразделе буду рассмотрены популярные построчные СУБД, которые могут быть использованы для реализации хранения в разрабатываемом программном продукте.

PostgreSQL

PostgreSQL [13] – это свободно распространяемая объектно-реляционная система управления базами данных, наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных [14].

PostgreSQL предоставляет транзакции со свойствами атомарности, согласованности, изоляции, долговечности (ACID [15]), автоматически обновляемые представления, материализованные представления, триггеры, внешние ключи и хранимые процедуры. Данная СУБД предназначена для обработки ряда рабочих нагрузок, от отдельных компьютеров до хранилищ данных или веб-сервисов с множеством одновременных пользователей.

Рассматриваемая СУБД управляет параллелизмом с помощью технологии управления многоверсионным параллелизмом (англ. MVCC [16]). Эта технология дает каждой транзакции «снимок» текущего состояния базы

данных, позволяя вносить изменения, не затрагивая другие транзакции. Это в значительной степени устраняет необходимость в блокировках чтения (англ. read lock [17]) и гарантирует, что база данных поддерживает принципы ACID.

Oracle Database

Oracle Database [18] – объектно-реляционная система управления базами данных компании Oracle [19]. На данный момент, рассматриваемая СУБД является самой популярной в мире. [20]

Все транзакции Oracle Database соответствуют обладают свойствами ACID, поддерживает триггеры, внешние ключи и хранимые процедуры. Данная СУБД подходит для разнообразных рабочих нагрузок и может использоваться практически в любых задачах. Особенностью Oracle Database является быстрая работа с большими массивами данных.

Oracle Database может использовать один или более методов параллелизма. Сюда входят механизмы блокировки для гарантии монопольного использования таблицы одной транзакцией, методы временных меток, которые разрешают сериализацию транзакций и планирование транзакций на основе проверки достоверности.

MySQL

MySQL [21] – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle.

Рассматриваемая СУБД имеет два основных движка хранения данных: InnoDB [22] и myISAM [23]. Движок InnoDB полностью полностью совместим с принципами ACID, в отличии от движка myISAM. СУБД MySQL подходит для использования при разработке веб-приложений, что объясняется очень тесной интеграцией с популярными языками PHP [24] и Perl [25].

Реализация параллелизма в СУБД MySQL реализовано с помощью механизма блокировок, который обеспечивает одновременный доступ к данным.

1.4.4 Выбор СУБД для решения задачи

Для решения задачи была выбрана СУБД PostgreSQL, потому что данная СУБД имеет поддержку языка plpython3u [26], который упрощает процесс интеграции базы данных в разрабатываемое приложение. Кроме того, PostgreSQL проста в развертывании.

1.5 Кэширование данных

Для ускорения быстродействия разрабатываемого приложения, можно прибегнуть к кэшированию данных. Для кэширования данных можно использовать NoSQL [27] in-memory базы данных. Такие базы данных хранят данные в оперативнгой памяти, что обеспечивает более быстрый доступ к данным.

1.5.1 Проблемы кэширования данных

Синхронизация данных

Приложение пишет в кэш, и в базу, которые между собой никак не реплицируются. Таким образом возникает несогласованность данных. Например, в случае разрабатываемого приложения, возможна ситуация, когда данные удаляются из хранилища и их нужно удалить из кэша. Эту проблему можно решить установкой тригеров в базе данных хранения рабочих программ дисциплин, которые буду срабатывать на изменение / удаление данных и синхронизировать актуальные данные в кэше.

Проблема «холодного старта»

Когда кэш только развертывается, он пуст и в нем нет никаких данных. Все запросы идут напрямую в базу данных, и только спустя какое-то время кэш будет «разогрет» и будет работать в полную силу. Эту проблему можно решить, выбрав СУБД с журналированием всех операций: при

перезагрузке можно восстановить предыдущее состояние кэша с помощью журнала событий, который хранится на диске. При этом, при перезапуске кэша, нужно синхронизировать данные с хранилищем: возможно, какие-то данные находящиеся в кэше перестали быть актуальными за время его перезагрузки.

1.5.2 Обзор in-memory NoSQL СУБД

Tarantool

Tarantool [28] – это платформа in-memory вычислений с гибкой схемой хранения данных для эффективного создания высоконагруженных приложений. Включает себя базу данных и сервер приложений на языке программирования Lua [29].

Tarantool обладает высокой скоростью работы по сравнению с традиционными СУЬД. При этом, в рассматриваемой платформе для транзакций реализованы свойства ACID, репликация master-slave [30] и master-master [31], как и в традиционных СУБД.

Для хранения данных используется кортежи (англ. tuple) данных. Кортеж – это массив не типизированных данных. Кортежи объединяются в спейсы (англ. space), аналоги таблицы из реляционной модели хранения данных. Спейс – коллекция кортежей, кортеж – коллекция полей.

В рассматриваемой СУБД реализованы два движка хранения данных: memtx [32] и vinyl [32]. Первый хранит все данные в оперативной памяти, а второй на диске. Для каждого спейса можно задавать различный движок хранения данных.

Каждый спейс должен быть проиндексирован первыичным ключом. Кроме того, поддерживается неограниченное количество вторичных ключей. Каждый из ключей может быть составным.

В Tarantool реализован механизм «снимков» текущего состояния хранилища и журналирования всех операций, что позволяет восстановить состояние базы данных после ее перезагрузки.

Redis

Redis [33] – резидентная система управлениями базами данных класса NoSQL с открытым исходным кодом. Основной структурой данных, с которой работает Redis является структура типа «ключ-значение». Данная СУБД используется как для хранения данных, так и для реализации кэшей и брокеров сообщений.

Redis хранит данные в оперативной памяти и снабжена механизмом «снимков» и журналирования, что обеспечивает постоянное хранение данных. Предоставляются операции для реализации механизма обмена сообщениями в шаблоне «издатель-подписчик»: с его помощью приложения могут создавать программные каналы, подписываться на них и помещать в эти каналы сообщения, которые будут получены всеми подписчиками. Существует поддержка репликации данных типа master-slave, транзакций и пакетной обработки комманд.

Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Ключевое отличие Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками. Поддерживаются следующие абстрактные типы данных:

- строки;
- списки;
- множества;
- хеш-таблицы;
- упорядоченные множества.

Тип данных значения определяет, какие операции доступные для него; поддежриваются высокоуровневые операции: например, объединение, разность или сортировка наборов.

1.5.3 Выбор СУБД для решения задачи

Для кэширования данных была выбрана СУБД Tarantool, так как она проста в развертывании и переносимости, и имеет подходящие коннекторы для базы данных PostgreSQL.

1.6 Формализация данных

1.6.1 База данных рабочих программ дисциплин

1.6.2 База данных кэшируемой информации

База данных кэшируемых значений должна хранить значения без дополнительной обработки. Данные должны быть актуальны и синхронизированны с основным хранилищем: кэш должен обновляться после каждой транзакции. Кроме того, нужно ограничить размер кэша и добавить вытеснение из него, например, с помощью политики вытеснения LRU [34] (Last Recently Used).

Вывод

В данном разделе:

- рассмотрена структура рабочей программы дисциплины и выявлены её наиболее интересные части;
- проанализированы способы хранения информации для система и выбраны оптимальные способы для решения поставленной задачи;
- проведен анализ СУБД, используемых для решения задачи и также выбраны оптимальные информационные системы;

- рассмотрена проблема актуальности кэшируемых данных и предложенно ее решение;
- формализованны данные, используемые в системе.

2 Конструкторская часть

2.1 Проектирование отношений сущностей

2.2 Проектирование базы данных рабочих программ дисциплин

2.3 Проектирование базы данных кэширования

Вывод

3 Технологическая часть

3.1 Архитектура приложения

3.2 Средства реализации

3.3 Детали реализации

3.4 ? REST ?

Вывод

4 Исследовательская часть

4.1 Постановка эксперимента

4.1.1 Цель эксперимента

4.1.2 Описание эксперимента

4.1.3 Результат эксперимента

Вывод

Заключение

Литература

[1] Положение о порядке разработки и утверждение рабочей программы дисциплины [Электронный ресурс]. Режим доступа: https://www.volgmed.ru/uploads/files/2010-11/1180-polozhenie_o_poryadke_razrabotki_i_utverzhdeniya_rabochej_programmy_uchebnoj_discipliny_(kursa).doc (дата обращения: 07.06.2021).

[2] Learning Management System (LMS) — HSE [Электронный ресурс]. Режим доступа: https://www.hse.ru/en/studyspravka/lms_student/ (дата обращения: 07.06.2021).

[3] Microsoft Word - Word Processing Software | Microsoft 365 [Электронный ресурс]. Режим доступа: https://www.microsoft.com/en-us/microsoft-365/word (дата обращения: 07.06.2021).

[4] What is a REST API? - Red Hat [Электронный ресурс]. Режим доступа: https://www.redhat.com/en/topics/api/what-is-a-rest-api (дата обращения: 07.06.2021).

[5] Образовательные стандарты | МГТУ им. Н. Э. Баумана [Электронный ресурс]. Режим доступа: https://bmstu.ru/plain/eduStandarts/ (дата обращения: 07.06.2021).

[6] МГТУ им. Н. Э. Баумана – Официальный сайт [Электронный ресурс]. Режим доступа: https://bmstu.ru/ (дата обращения: 07.06.2021).

[7] Что такое база данных | Oracle Россия и СНГ [Электронный ресурс]. Режим доступа: https://www.oracle.com/ru/database/what-is-database/ (дата обращения: 07.06.2021).

[8] Что такое СУБД - RU-CENTER [Электронный ресурс]. Режим доступа: https://www.nic.ru/help/chto-takoe-subd_8580.html (дата обращения: 07.06.2021).

[9] Что такое микросервисная архитектура: простое объяснение | MCS Mail.ru [Электронный ресурс]. Режим доступа: https://mcs.mail.ru/blog/prostym-jazykom-o-mikroservisnoj-arhitekture (дата обращения: 07.06.2021).

[10] Race conditions and deadlocks - Microsoft Docs [Электронный ресурс]. Режим доступа: https://docs.microsoft.com/en-us/troubleshoot/dotnet/visual-basic/race-conditions-deadlocks (дата обращения: 07.06.2021).

[11] What is OLTP? | IBM [Электронный ресурс]. Режим доступа: https://www.ibm.com/cloud/learn/oltp (дата обращения: 07.06.2021).

[12] What is OLAP? | IBM [Электронный ресурс]. Режим доступа: https://www.ibm.com/cloud/learn/olap (дата обращения: 07.06.2021).

[13] PostgreSQL: Документация. [Электронный ресурс]. Режим доступа: https://postgrespro.ru/docs/postgresql/ (дата обращения: 07.06.2021).

[14] PostgreSQL: вчера, сегодня, завтра [Электронный ресурс]. Режим доступа: https://postgrespro.ru/blog/media/17768 (дата обращения: 07.06.2021).

[15] Транзакции, ACID, CAP | GeekBrains [Электронный ресурс]. Режим доступа: https://gb.ru/posts/acid_cap_transactions (дата обращения: 07.06.2021).

[16] Documentation: 12: 13.1. Introduction - PostgreSQL [Электронный ресурс]. Режим доступа: https://www.postgresql.org/docs/12/mvcc-intro.html (дата обращения: 07.06.2021).

[17] Применение блокировок чтения/записи | IBM [Электронный ресурс]. Режим доступа: https://www.ibm.com/docs/ru/aix/7.2?topic=programming-using-readwrite-locks (дата обращения: 07.06.2021).

[18] SQL Language | Oracle[Электронный ресурс]. Режим доступа: https://www.oracle.com/database/technologies/appdev/sql.html (дата обращения: 07.06.2021).

[19] Oracle | Integrated Cloud Applications and Platform Services [Электронный ресурс]. Режим доступа: https://www.oracle.com/index.html (дата обращения: 07.06.2021).

[20] DB-Engines Ranking [Электронный ресурс]. Режим доступа: https://db-engines.com/en/ranking (дата обращения: 07.06.2021).

[21] MySQL Database Service is a fully managed database service to deploy cloud-native applications. [Электронный ресурс]. Режим доступа: https://www.mysql.com/ (дата обращения: 07.06.2021).

[22] MySQL Reference Manual 8.0: The InnoDB Storage Enginee [Электронный ресурс]. Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/innodb-storage-engine.html (дата обращения: 07.06.2021).

[23] MySQL Reference Manual 16.2: The MyISAM Storage Engine [Электронный ресурс]. Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/myisam-storage-engine.html (дата обращения: 07.06.2021).

[24] PHP: Hypertext Preprocessor [Электронный ресурс]. Режим доступа: https://www.php.net/ (дата обращения: 07.06.2021).

[25] The Perl Programming Language [Электронный ресурс]. Режим доступа: https://www.perl.org/ (дата обращения: 07.06.2021).

[26] PostgreSQL: Документация: 9.6: 44.1. Python 2 и Python 3. [Электронный ресурс]. Режим доступа: https://postgrespro.ru/docs/postgresql/9.6/plpython-python23 (дата обращения: 07.06.2021).

[27] NoSQL базы данных: понимаем суть / Хабр [Электронный ресурс]. Режим доступа: \T2A\CYRCH\T2A\cyrt\T2A\cyro\T2A\cyrt\T2A\cyra\T2A\cyrk\T2A\cyro\T2A\cyreNoSQL?|AmazonAWS (дата обращения: 07.06.2021).

[28] Tarantool – Платформа In-memory вычислений [Электронный ресурс]. Режим доступа: https://www.tarantool.io/ru/ (дата обращения: 07.06.2021).

[29] The Programming Language Lua [Электронный ресурс]. Режим доступа: http://www.lua.org/ (дата обращения: 07.06.2021).

[30] Tech Confronts Its Use of the Labels «Master» and «Slave» [Электронный ресурс]. Режим доступа: https://www.wired.com/

story/tech-confronts-use-labels-master-slave/ (дата обращения: 07.06.2021).

[31] How To Set Up MySQL Master-Master Replication [Электронный ресурс]. Режим доступа: https://www.digitalocean.com/community/tutorials/how-to-set-up-mysql-master-master-replication (дата обращения: 07.06.2021).

[32] Движки базы данных | Tarantool [Электронный ресурс]. Режим доступа: https://www.tarantool.io/ru/doc/latest/book/box/engines/ (дата обращения: 07.06.2021).

[33] Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker [Электронный ресурс]. Режим доступа: https://redis.io/ (дата обращения: 07.06.2021).

[34] LRU, метод вытеснения из кэша | Habr [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/136758/ (дата обращения: 07.06.2021).