Федеральное государственное автономное образовательное учреждение
высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики им. А.Н Тихонова

Департамент компьютерной инженерии

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННОЙ БАЗЫ ДАННЫХ

Методические указания

по выполнению практического задания

по курсу "Распределенные базы данных и сетевые вычисления"

Часть 1: Распределенные базы данных

Москва

2023

Составители к.т.н., доцент И. П. Карпова,
к.ф.-м.н., доцент Чернышов Л. Н.

УДК 681.3

Проектирование и реализация распределенной базы данных: Методические указания по выполнению практического задания по курсу "Распределенные базы данных и сетевые вычисления", часть 1: "Распределенные базы данных", 1-2 модуль. / Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ; Сост.: И.П. Карпова, Л.Н. Чернышов. – М., 2023. – 39 с.

Практическое задание посвящено реализации проекта распределенной базы данных (РБД), особое внимание уделено использованию таких методов поддержки распределенности, как репликация и консолидация данных.

Для студентов магистратуры технических факультетов вузов, изучающих распределенные автоматизированные информационные системы и системы управления распределенными базами данных.

Табл. 1. Ил. 10. Библиогр.: 10 назв.

СОДЕРЖАНИЕ

1. Цель работы

На практических занятиях необходимо реализовать проект распределенной БД, созданный в 1-м модуле в ходе выполнения домашнего задания, с целью применения на практике знаний, полученных в процессе изучения курса "Распределенные базы данных". Получение практических навыков создания распределенной автоматизированной информационной системы (РАИС), позволит проверить работоспособность проекта РБД, созданного в 1-м модуле, и получить квалификацию программиста, занимающегося реализацией прикладной логики РАИС.

2. Общие положения

Проектирование распределенной базы данных (РБД) является одной из наиболее сложных и ответственных задач, связанных с созданием РАИС.

Проектирование базы данных – это процесс, который подразумевает использование определённой технологии. Никто не сомневается в том, что в случае нарушения технологии изготовления печатной платы, например, эта плата либо вообще не будет работать, либо не будет соответствовать заявленным характеристикам. Но почему-то считается, что соблюдать технологию проектирования БД (и вообще программного обеспечения) совершенно необязательно. И начинают работу по реализации реляционной БД с создания таблиц. Получившаяся в ходе такого "проектирования" база данных будет ненадёжной, неэффективной и сложной в сопровождении. (Исключением могут быть случаи простых предметных областей, которые можно отразить в реляционной базе данных, состоящей из 3-4 таблиц). Поэтому при создании базы данных необходимо придерживаться правил технологии проектирования БД. Если, конечно, вас интересует результат.

Подробно процесс проектирования реляционной базы данных изложен в [1-3]. В результате вы должны получить схему БД и схемы отношений (таблиц), приведенные к 4-й нормальной форме (4НФ). Далее с учетом особенностей вашей предметной области вы должны предложить схему фрагментации этой БД и методы поддержки распределенности, обеспечивающие основные потребности пользователей вашей БД. Пример разработки схемы фрагментации и обоснование выбора методов поддержки распределенности приведен в Приложении А.

3. Реализация проекта РБД

Перед созданием РБД необходимо установить соответствующее программное обеспечение (СУБД и вспомогательные средства) и настроить их определенным образом. В данном пособии приведено два варианта настройки: с использованием технологии виртуализации (раздел 3.1) и без виртуали-

зации (Приложение Б). Список источников для настройки репликации в СУБД PostgreSQL приведен в Приложении Г.

3.1. Установка и настройка MySQL для работы в режиме master/slave реплики

Далее описывается установка и настройка последней (на момент написания данного пособия) версии СУБД MySQL 8.0 на ОС Linux Debian. Для удобства развертывания нескольких баз данных будем использовать такие технологии, как Docker и Docker Compose.

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему.

Docker Compose – технология удобной настройки и управления приложениями, развернутыми в Docker.

На операционные системы Linux, MacOS докер ставится “из коробки”, для ОС Windows используется программа Docker Desktop (только для Windows 10 Pro) или с использованием WSL [4]. В данном руководстве будем использовать ОС Windows и WSL [5].

Создадим в любом желаемом месте отдельную папку с именем mysql-docker. Все дальнейшие манипуляции будем проводить в ней (Рис. 1).

Рис. 1. Папка mysql-docker

Создадим в этой папке файл *master.cnf* для настройки работы master-сервера СУБД.

```
[mysqld]
server-id=1 # id сервера для идентификации
log-bin     # способ хранения информации об изменении на master
# Параметры, уже присутствующие в MySQL после установки
# Файл, в который будут записаны id процессов, созданных СУБД
pid-file    = /var/run/mysqld/mysqld.pid
# Файл через который осуществляется сокетное (TCP) соединение
    между серверами СУБД или клиентским терминалом
socket      = /var/run/mysqld/mysqld.sock
# Директория, где хранятся все базы данных и данные внутри них
datadir     = /var/lib/mysql
# Параметр определяет, из какой директории можно свободно скачи-
    вать данные БД (в данном случае скачивать вообще нельзя)
secure-file-priv = NULL
```

Листинг 1. Файл master.cnf

Log-bin параметр указывает СУБД, что для регистрации любых изменений в базе (добавление записей, создание таблиц и т.д.) необходимо использовать двоичный журнал обновлений. Это способ записи событий и времени их наступления в СУБД в виде логов в бинарном формате [6].

Создадим в папке файл slave.cnf для настройки работы slave-сервера СУБД.

```
[mysqld]
server-id=2 # id сервера для идентификации
log-bin
pid-file  = /var/run/mysqld/mysqld.pid
socket    = /var/run/mysqld/mysqld.sock
datadi    = /var/lib/mysql
secure-file-priv = NULL
binlog-do-db = librarydb # указываем какую БД реплицируем
```

Листинг 2. Файл slave.cnf

Также для работы master/slave режима необходимо прописать sql-файл для создания пользователя, от имени которого будет происходить репликация, с полномочиями для репликации.

```
CREATE USER repl@'%' IDENTIFIED WITH mysql_native_password BY
'password';
GRANT REPLICATION SLAVE ON *.* TO repl@'%';
```

Листинг 3. Файл master.sql

Данная команда создает пользователя repl (запись “@’%’” после имени пользователя означает, что slave-сервер может находиться на любом хосте) с паролем «password», указывая при этом, что алгоритм аутентификации будет «mysql_native_password». Этот алгоритм используется для аутентификации по умолчанию и применяет сравнение паролей по хешам формата SHA-1. Далее командой GRANT REPLICATION SLAVE пользователю repl назначаются права на репликацию любой таблицы/схемы/базы (ON *.*).

Для перенастройки slave-server на получение данных от master-server будем использовать следующий скрипт:

```
CHANGE MASTER TO MASTER_HOST='mysql-master', MASTER_USER='repl',
MASTER_PASSWORD='password';
```

Листинг 4. Файл slave.sql

Данная команда сообщает СУБД, что она должна использовать в качестве основного сервера тот, который указан в параметре MASTER_HOST (передается хост мастера: для докера – это имя контейнера, т.к. используется DNS). Также для подключения передается имя пользователя, у которого есть

права на репликацию (repl), и пароль. Все основные подготовительные работы завершены, и теперь можно написать docker-compose файл, который скачает из публичного докер-репозитория (DockerHub) образ СУБД MySQL (путь определяется по имени образа в compose-файле), проставит необходимые переменные окружения (например, пароль root-пользователя), а также передаст в контейнер все заготовленные нами конфигурационные и sql-скрипты.

Полная версия docker-compose.yml файла представлена в листинге 5.

```
# Версия интерпретатора Python (указываем и забываем)
version: "3.9"
# Описание всех серверов MySQL
services:
  # Описание master-server
  mysql-master:
    # Имя образа MySQL СУБД, скачиваемый из DockerHub (официальный образ)
    image: 'mysql:8.0'
    # Имя докер-контейнера (имя запущенного докер-образа, также имя хоста в DNS)
    container_name: mysql-master
    # Передача файлов из местной ОС внутрь контейнера
    volumes:
      - ./master.cnf:/etc/mysql/my.cnf
      - ./master.sql:/docker-entrypoint-initdb.d/start.sql
    # Указываем переменную окружения внутри контейнера
    environment:
      MYSQL_ROOT_PASSWORD: "password"
    # Пробрасываем порт с хостовой машины внутрь докера для возможности подключения к СУБД снаружи
    ports:
     - '3306:3306'
  # Описание Slave-server
  mysql-slave:
    image: 'percona:8.0'
    container_name: mysql-slave
    volumes:
      - ./slave.cnf: /etc/mysql/my.cnf
      - ./slave.sql:/docker-entrypoint-initdb.d/start.sql
    # Указываем, что slave запускается только после успешного запуска master
    depends_on:
      - mysql-master
    environment:
      MYSQL_ROOT_PASSWORD: "password"
```

```
ports:
 - '3307:3306'
```

Листинг 5. Файл docker-compose.yml

ВАЖНО: все созданные конфигурационные файлы должны иметь права доступа на чтение и исполнение (chmod 555). В противном случае MySQL не станет их читать и будет использовать настройки по умолчанию.

Запускаем *docker-compose* файл командой *docker-compose up* (Рис. 2).

```
Creating network "mysql_default" with the default driver
Creating mysql-master ... done
Creating mysql-slave  ... done
```

Рис. 2. Успешный запуск docker-compose

Для тестирования master/slave репликации будем использовать командную строку каждого контейнера. Для этого вызываем команду *docker ps* (показывает список всех запущенных контейнеров), чтобы узнать имя нужного контейнера (Рис. 3).

```
jvvladimir@DESKTOP-VHF9DNR:~/mysql-docker$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS             PORTS                                NAMES
66cf2f808c29   percona:8.0    "/docker-entrypoint.…"   2 hours ago   Up About a minute  33060/tcp, 0.0.0.0:3307->3306/tcp    mysql-slave
579536f61a19   percona:8.0    "/docker-entrypoint.…"   2 hours ago   Up About a minute  0.0.0.0:3306->3306/tcp, 33060/tcp    mysql-master
```

Рис. 3. Команда docker ps

Далее запускаем команду *docker exec -it mysql-master bash* (открываем bash для контейнера по имени) (Рис. 4).

```
jvvladimir@DESKTOP-VHF9DNR:~$ docker exec -it mysql-master bash
bash-4.4$
```

Рис. 4. Терминал докер-контейнера

Далее необходимо зайти под root пользователем (*mysql -uroot -ppassword*) в MySQL для создания новой БД (Рис. 5).

```
bash-4.4$ mysql -uroot -ppassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.22-13 Percona Server (GPL), Release 13, Revision 6f7822f

Copyright (c) 2009-2020 Percona LLC and/or its affiliates
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Рис. 5. Терминал MySQL

Создаем новую БД и проверяем, что она появилась (Рис. 6).

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
5 rows in set (0.00 sec)

mysql> create database library;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| library            |
| mysql              |
| performance_schema |
| sys                |
| test               |
+--------------------+
6 rows in set (0.00 sec)
```

Рис. 6. Создание новой БД

Для проверки master/slave репликации необходимо проделать тот же самый набор действий для контейнера, в котором находится slave, и убедиться, что любые изменения в master (создание баз, таблиц, добавление записей) мгновенно отражаются и в slave-сервере.

Для более наглядного представления рекомендуется использовать внешние клиенты к СУБД (DataGrip, DBeaver и др.). Пример подключения через DataGrip приведен ниже (Рис. 7).

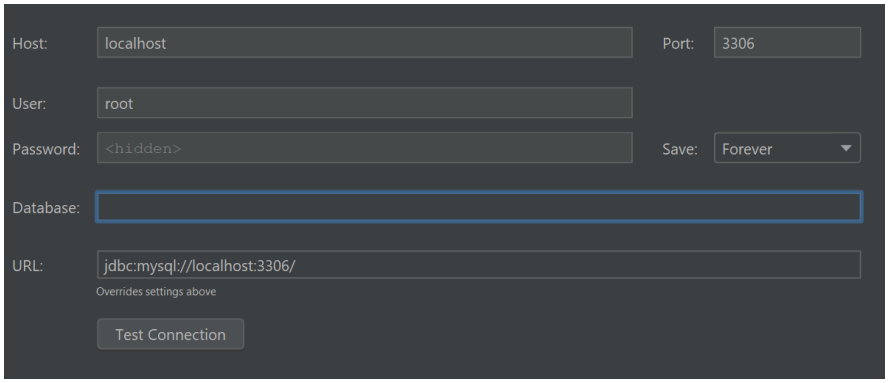

Рис. 7. Подключение через внешний клиент

Подключение к master и slave серверам будет отличаться только указанным портом: master:3306, slave:3307.

3.2. Настройка репликации без основной копии с асинхронным распространением изменений

Настройка описывается для проекта, представленного в Приложении А.

В данном проекте есть только одна таблица, которую необходимо реплицировать без основной копии асинхронно и для master, и для slave – таблица читателей. Технически это означает, что master теперь сам будет репликой для slave, но только в рамках одной таблицы.

Для настройки такого поведения доработаем файл master.cnf, листинг которого указан ниже:

```
[mysqld]
server-id=1
log-bin
pid-file    = /var/run/mysqld/mysqld.pid
socket     = /var/run/mysqld/mysqld.sock
datadir     = /var/lib/mysql
secure-file-priv = NULL
replicate-do-table=librarydb.reader # указываем какую конкретно таблицу реплицируем, остальные будут игнорироваться
binlog-do-db = librarydb
```

Листинг 6. Файл master.cnf

Здесь мы специально указываем, какую таблицу будем реплицировать из slave. Далее необходимо доработать файл master.sql и slave.sql по образу и подобию того, как это делали в slave.

Создаем пользователя, от имени которого будет происходить репликация, с полномочиями для репликации:

```
CREATE USER replslave@'%' IDENTIFIED WITH mysql_native_password BY 'password';

GRANT REPLICATION SLAVE ON *.* TO replslave@'%';
```

Листинг 7. Файл slave.sql

Для перенастройки master-server на получение данных от slave-server будем использовать следующий скрипт:

```
CHANGE MASTER TO MASTER_HOST='mysql-slave, MASTER_USER='replslave', MASTER_PASSWORD='password';
```

Листинг 8. Файл master.sql

После настройки в клиенте для СУБД можем заметить, что при добавлении новых записей в таблицу читателей, данные видны как в master, так и в slave, независимо от того, на каком из серверов инициировалось это добавление.

3.3. Настройка консолидации

Для настройки консолидации в Docker необходимо установить еще один сервер MySQL с названием mysql-consol. Docker compose файл приведен ниже:

```
# Описание Consol-server
  mysql-consol:
    image: 'mysql:8.0'
    cap_add:
      - SYS_NICE
    container_name: mysql-consol
    volumes:
      - ./consol.cnf:/etc/mysql/my.cnf
      - ./consol.sql:/docker-entrypoint-initdb.d/start.sql
    # Указываем, что consol запускается только после успешного запуска master и slave
    depends_on:
      - mysql-master
      - mysql-slave
    environment:
      MYSQL_ROOT_PASSWORD: "password"
    ports:
     - '3308:3306'
```

Листинг 9. Файл docker-compose.yml

Задача данного сервера – подключиться к master и slave серверам и считывать данные одновременно с двух таблиц, реплицируя их у себя. Еще до реализации данного решения очевидны две большие проблемы. Первая из них заключается в том, каким образом будет проходить репликация записей из двух разных мест в том случае, если первичные ключи записей совпадают.

MySQL не может сам разрешить данную проблему (например, автоматическим изменением первичного ключа записи, если запись с таким ключом уже есть). Для решения данной проблемы были найдены четыре возможных варианта:

1) Реплицировать таблицы без первичных ключей вообще.
2) Сделать ключ в реплицируемых таблицах составным, чтобы пара значений всегда была уникальной (например, id записи и id узла – с нашем случае, идентификатор библиотеки).
3) Реплицировать данные не с помощью MySQL, а программно через код (самостоятельно по какому-либо алгоритму проводить агрегацию данных).
4) Задать для разных узлов разные диапазоны значений ПК.

Вторая проблема связана с внешними ключами. Реплицируемые таблицы могут иметь взаимосвязи с другими таблицами, однако в консолидации этих таблиц нет. Если мы хотим консолидировать только конкретные таблицы, тогда необходимо удалить все внешние ключи на отсутствующие у нас таблицы (убрать ограничение целостности foreign key, а не сами поля из таблиц).

По аналогии с master и slave серверами создадим в папке mysql-docker файл consol.cnf. Листинг файла приведен ниже:

```
[mysqld]
server-id=3
log-bin
pid-file  = /var/run/mysqld/mysqld.pid
socket    = /var/run/mysqld/mysqld.sock
datadir   = /var/lib/mysql
secure-file-priv= NULL
# Указываем таблицы, которые собираемся реплицировать
replicate-do-table=librarydb.orders
replicate-do-table=librarydb.accounting
replicate-do-table=librarydb.books_in_library
binlog-do-db = librarydb
```

Листинг 10. Файл conf.cnf

В данном конфигурационном файле мы указываем, какие таблицы из всей базы собираемся консолидировать, остальные нас не интересуют. Далее создадим файл consol.sql и пропишем в нем репликацию сразу для двух сервером: master и slave. Это делается с помощью специальной абстракции, называемой «каналом». К каждому каналу подключается пользователь для репликации, которого мы уже создавали на предыдущих шагах. Далее репликация двух каналов запускается, и начинается процесс консолидации данных по трем таблицам.

```
CHANGE MASTER TO MASTER_HOST='mysql-master', MASTER_USER='repl',
MASTER_PASSWORD='password' FOR CHANNEL "master";
CHANGE MASTER TO MASTER_HOST='mysql-slave', MAS-
TER_USER='replslave', MASTER_PASSWORD='password' FOR CHANNEL
"slave";
START SLAVE FOR CHANNEL "master";
START SLAVE FOR CHANNEL "slave";
```

Листинг 11. Файл consol.sql

ВАЖНО: не забудьте про то, что для mysql-consol сервера необходимо создавать только те таблицы, которые нужно консолидировать. Удаляйте все внешние ключи на таблицы, которых нет в консолидированной копии.

4. Реализация проекта распределенной информационной системы

4.1. Общие положения

На практических занятиях 2-го модуля необходимо реализовать распределенную автоматизированную информационную систему (РАИС) на основе спроектированной в 1-м модуле распределенной БД. В процессе реализации системы студенты должны использовать навыки разработки информационных систем, которые они получали при учебе в бакалавриате. Предполагается, что они освоили программирование на одном или нескольких языках программирования, имеют навыки использования соответствующего инструментального ПО, имеют понятие о моделях жизненного цикла программных систем, знают требования ГОСТ на программную документацию.

Реализация РАИС в полном объеме может оказаться трудоемкой задачей, что не позволит ее выполнить в отведенное время. В таком случае допускается реализация лишь части функций, предусмотренных предварительным техническим заданием. Также допустимо упрощение информационной модели (сокращение числа сущностей) и упрощение сценариев работы для ролей пользователей системы. Такое подмножество должно быть определено в начале работы и согласовано с преподавателем.

Обязательное требование к выбранному подмножеству системы – наличие частей, поддерживающих методы репликации данных и распределенных запросов или транзакций – в зависимости от того, что входит в проект РБД.

Предлагается выбрать стандартную модель жизненного цикла с основными этапами работ:

- техническое задание (ТЗ) (анализ и определение требований);
- технический проект;
- рабочий проект.

Ограничение функций в подмножестве системы в основном касается рабочего проекта. Технический проект может описывать систему в полном объеме.

В 1-м модуле разработка системы в основном касалась проектирования информационной модели и в меньшей степени бизнес-процессов (сценарии работ). Это охватывало большую часть ТЗ, а также частично технического и рабочего проектов в информационном плане. На следующем этапе необходимо доработать как ТЗ, так и описать и реализовать технический и рабочий проекты.

Работа, как правило, выполняется группой студентов – бригадой. Отдельные части выполняемых работ распределяются между членами бригады.

В итоге оценивается как полученный общий результат – реализованная РА-ИС, так и вклад каждого участника.

Результат работы представляется в форме итогового отчета и программного кода самой системы. Разделы отчета (см. Приложение Б) примерно соответствуют разделам стандартных документов ЕСПД [7], с которыми студентам необходимо ознакомиться.

Очень важен для успешной работы выбор программной платформы, подходящих фреймворков и инструментальных средств. Здесь нет единого требования – каждая бригада делает самостоятельный выбор. Для СУБД рекомендуется либо MySQL [6], либо PostgreSQL [полезные ссылки, 9]. В качестве языка программирования для клиентских компонент – язык Python в составе платформы Django [полезные ссылки, 9], либо языки HTML и JavaScript (клиент Web-браузер). Для описания артефактов технического проекта желательно использовать язык UML [полезные ссылки, 5].

4.2. Содержание работ

Основная цель всей работы – освоить технологию проектирования и реализации распределенных систем. В итоговой версии системы (или ее подмножества) должны быть реализованы те сценарии работы, в результате которых в распределенной БД происходят процессы репликации и выполнения распределенных запросов и/или транзакций. После заполнения баз данных и настройки распределенной БД, выполнение действий в каждом экземпляре БД можно осуществлять непосредственно из панелей администрирования. Эти действия должны выполняться в форме заранее подготовленных команд SQL. Этому необходимо научиться на начальных этапах работы. Но в дальнейшем эти действия должны инициироваться из экранных форм конечных пользователей.

Поскольку разработка интерфейсов пользователей не является важной задачей при изучении данной дисциплины, требования к интерфейсу могут быть не столь строгими. Например, можно ограничиться простыми стилями оформления интерфейсов, не проводить контроль данных при вводе и т.п. Совсем необязательным является компонент входа в систему с аутентификацией и авторизацией пользователя.

В обычных случаях разработка систем ведется последовательно по этапам: сначала разрабатывается ТЗ, потом технический проект, потом рабочий проект. В нашем случае, когда уже имеется ТЗ и проект РБД, части этапов можно выполнять параллельно, распределяя их между участниками бригады.

Далее уточняются требования к работам на разных этапах.

Техническое задание. В предварительных требованиях, которые были сформулированы на этапе разработки информационной модели, следует внимательно проработать те функции, которые предполагается реализовать в

первую очередь, т. е. в выбранном подмножестве. К описанию предметной области, ролей пользователей и функциональным требований следует добавить описание нефункциональных требований. Наиболее важные из них, которые могут повлиять на выбор архитектуры и технологий реализации, – это объемы данных, многопользовательский режим (частота обращений к системе), требования к временным характеристикам, квалификации пользователей.

Технический проект. Самое важное решение технического проекта – выбор архитектуры системы. В общем случае выбор зависит от требований ТЗ. Но для реализации подмножества системы, в котором демонстрируется возможности технологии распределенных данных, можно ограничиться самой простой архитектурой клиент-сервер. При этом необходимо определить, где будет реализовываться логика системы, т.е. будет ли клиент "тонким" или "толстым". Например, когда клиентом является веб-браузер, формирование экранной формы может производиться как на сервере, так и на клиенте.

Важной частью технического проекта является проектирование интерфейсов. В данном случае можно упростить требования и по возможности использовать готовые решения. Например, можно использовать библиотеки для работы с таблицами, которые есть в каждой системе программирования для разных языков.

Сценарии работы для основных процессов желательно описать с помощью UML-диаграмм. Необходимо обратить внимание не только на прохождение основных действия, но и действий, возникающих при нештатных ситуациях. UML-диаграммы можно создавать в системе StarUML [полезные ссылки, 10].

На этом этапе необходимо таже подготовить контрольные примеры, в которых будут описаны последовательности действий, которые продемонстрируют работоспособность распределенной системы. Должны быть продемонстрированы состояния баз данных как в начале, так и после выполнения действий, на которых будут видны результаты репликации (с основной копией, в том числе с консолидацией данных, или без основной копии) и результаты выполнения распределенных транзакций.

Рабочий проект. Первый шаг рабочего проекта – создание БД и заполнение ее данными. Для этого можно воспользоваться генераторами данных [4]. Количество данных должно быть достаточным, чтобы демонстрировать основные возможности системы.

Основную экранную форму входа в систему можно сделать в виде набора кнопок, каждая из которых открывает форму для пользователя соответствующей роли. Полноценный вход с вводом логина и пароля имеет смысл делать, только если есть готовое решение.

Элементы экранных форм для пользователя должны обеспечивать выполнение всех необходимых действий. При этом реализованы могут только действия, предусмотренные в выбранном подмножестве.

Необходимо предусмотреть включение отчетных форм, на которых, в частности, можно проверить результаты изменения состояния баз данных.

Основное требование к работающей системе – выполнение контрольного примера (примеров), который демонстрирует возможности распределенной системы.

При реализации интерфейса системы желательно выбирать самые простые решения, которые предоставляются в выбранной платформе. Примеры некоторых решений приводятся в следующем разделе.

Полученные артефакты на стадиях технического и рабочего проектов следует размещать в папке проекта.

4.3. Типовые решения для реализации компонент системы

В реальных проектах, как правило, сначала создается прототип, на котором проверяются основные требования. В дальнейшем с учетом замечаний заказчика реализация совершенствуется. Поэтому целесообразно в прототипе использовать типовые стандартные решения. Это особенно касается интерфейсов пользователя. Здесь приводятся примеры решений для web-приложений.

Для основных экранных форм стандартным интерфейсом является меню действий с выпадающими подменю (Рис. 8-9).

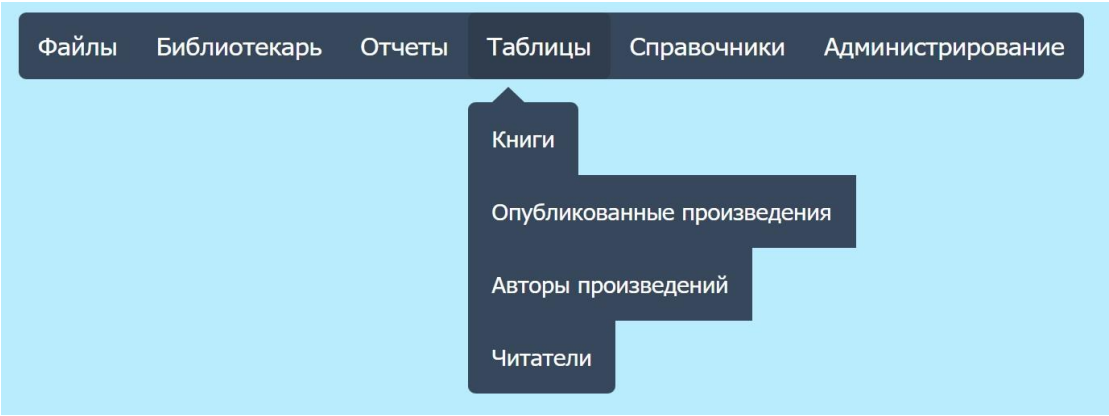

Рис. 8. Главное меню системы (таблицы)

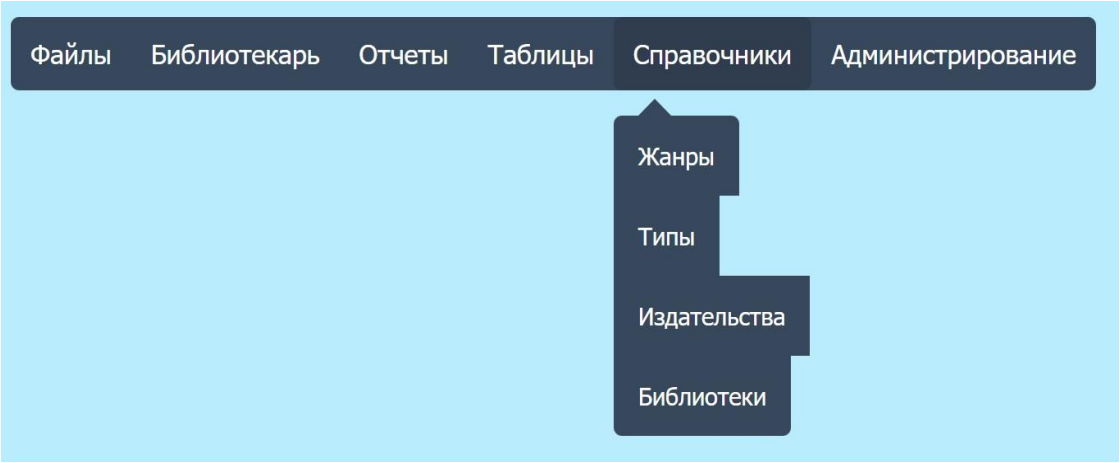

Рис. 9. Главное меню системы (справочники)

Формирование этого меню происходит на основе простого текстового файла, редактирование которого возможно в пункте «Администрирование»:

```
MENU Файлы
     Закончить_работу app_close.php
END
MENU Библиотекарь
     Выдача_книг
     Регистрация_читателя
END
MENU Таблицы
     Книги
     Опубликованные_произведения app-sprav.php?spr=spr-pr
     Авторы_произведений
     Читатели
END
. . .
```

После названия пункта меню указывается имя программы, которая будет вызываться при выборе пункта. Коды этого примера и следующих приводятся на Google-диске в папке «Примеры».

Наиболее разнообразны интерфейсы для просмотра/редактирования табличных данных. Библиотека Datatables [7] проста в использовании и имеет широкие возможности. Например, для простого просмотра, включая такие действия, как листание данных, сортировка по значениям колонок, поиск данных и др., достаточно разместить код, описывающий название колонок таблицы:

```
$(document).ready(function() {
     $('#example').DataTable( {
          "ajax": "libr1.json",
          columns: [
               { title: "Автор" },
               { title: "Название" },
               { title: "Жанр" },
               { title: "Код" },
```

```
                { title: "Файл" },
            ]
        } );
} );
```

Здесь данные выбираются из файла **"libr1.json"** в формате JSON. Экранная форма будет выглядеть так (Рис. 10):

Show 10 entries Search:

Автор	Название	Жанр	Код	Файл
Аксенов Василий	Апельсины из Марокко	драма	253195	apels
Аксенов Василий	Звездный билет	драма	321343	bilet
Аксенов Василий	Круглые сутки нон-стоп	драма	221538	non_stop
Аксенов Василий	Мой дедушка - памятник	детская	319415	aksenov1
Аксенов Василий	Остров Крым	драма	830944	krym
Аксенов Василий	Пора, мой друг, пора	драма	339712	pora
Аксенов Василий	Сундучок, в котором что-то стучит	сказка	287496	aksenov4
Акунин Борис	Азазель	драма	398832	azazel
Акунин Борис	Алтын-Толобас	драма	663917	akunin1
Акунин Борис	Декоратор	драма	313515	dekorat

Showing 1 to 10 of 26 entries Previous 1 2 3 Next

Рис. 10. вид таблицы в библиотеке Datatables

Предварительно может быть произведен запрос к БД, который сформирует необходимый json-файл.

Код, с помощью которого осуществляется доступ к базе данных, может быть примерно таким (**data_txt.php** – программа на языке PHP):

```
session_start();
if (!isset($_SESSION['count'])) {","");// связь с сервером
     $dbConn = mysql_connect("localhost","root
     if(!$dbConn){ echo "<h3>Нет соединения...<h3>\n"; return; }
     $_SESSION['dbconn'] = $dbConn;
} else {
     $dbConn = $_SESSION['dbconn'];
}
$tb  = $_REQUEST["file"];     // таблица БД
$do  = $_REQUEST["do"];       // действие
$key = $_REQUEST["key"];      // ключ
$val = $_REQUEST["val"];      // значение
$bd = "library";              // имя БД
$select = mysql_select_db($bd,$dbConn);      // связь с БД
if(!$select){ echo mysql_error() ."<br>\n"; return;}
$b = true;
if ($do=='del'){ // удалить
     $res = mysql_query(
```

```
        "DELETE FROM $tb WHERE id=$key",$dbConn);
     echo $res ? "запись id=$key удалена": mysql_error();

}else if($do=='change'){ // изменить
     $res = mysql_query(
        "UPDATE $tb SET name='".$val."' WHERE id=$key",$dbConn);
     echo $res ?"запись id=$key изменена" : mysql_error();
}else if($do=='add'){ // добавить
     $res = mysql_query(
        "INSERT INTO $tb VALUES ($key,'".$val."')",$dbConn);
     echo $res ? "запись id=$key добавлена": mysql_error();
}else if($do=='show'){ // отобразить
     $res = mysql_query(
            "SELECT * FROM $tb ORDER BY ID",$dbConn);
     if(!$res){echo "Ошибка:".mysql_error()."<br>\n";
               return $res;}
     echo "<table>";
     while ($row = mysql_fetch_assoc($res)){
          $x = $row['ID']; $y = $row['NAME'];
          echo "<tr><td class='td1'>$x<td class='td2'>$y";
     }
     echo "</table>";
}
```

В этом примере предполагается, что таблица имеет две колонки: ключ и значение. Названия колонок передаются из JavaScript-программы на клиенте. Также передается параметр «do», задающий действие: «del» – удалить запись, «change» – изменить данные, «add» – добавить запись, «show» – отобразить данные. В программе происходит вызов процедуры «mysql_query» с параметром соответствующей SQL-команды: «DELETE», «UPDATE», «INSERT» или «SELECT». Программа при первых трех случаях возвращает сообщение об успешном завершении или ошибке. В случае команды «show» возвращает HTML-код табличного представления.

На клиенте программа на языке JavaScript с использованием библиотеки JQuery осуществляет вызов серверной программы (для действий «show» и «add») :

```
     prog = "data_txt.php"
     function fshow(){ // показать таблицу
          $('#sprav').load(prog,{'do':'show','file':ft})
     }
     function fadd(){ // добавить строку
        var key = frm1.key.value
        var val = frm1.val.value
        $('#sprav')
        .load(prog,{'do':'add','file':ft, 'key':key, 'val':val})
     }
```

Данные для значений ключа и значения вводятся в форме «add» в полях с именами «key» и «val» соответственно:

```
<body onload='fshow()'>
<form name='frm1'>
     <button onclick="fadd()"> добавить <button>
     <button onclick="fdel()"> удалить  <button>
     <input name='key'>
     <input name='val'>
     <div id="sprav"></div>
</form>
```

Отображение таблицы происходит непосредственно при загрузке страницы в раздел с именем «sprav». Когда данные изменяются в результате действий добавления, удаления и изменения, необходимо обновить страницу.

Эти простые программы легко модифицируются, если таблицы имеют больше полей. А для стыковки с программой, использующей Datatables, в PHP-программе необходимо сформировать JSON-данные вместо таблицы.

В папке «Примеры» также могут быть размещены подобные примеры для других языков.

4.4. Организация и порядок выполнения работ

Реализация РАИС выполняется бригадой студентов (не более 4-х человек). В бригаде назначается ответственный исполнитель – бригадир. Распределение обязанностей и видов работ по этапам проводится бригадиром самостоятельно по согласованию с членами бригады.

Виды работ и сроки выполнения отражаются в таблице «Журнал работ» (Таблица 1), за ведение которой отвечает бригадир.

Таблица 1. Журнал работ

Этап работы	Дедлайн	Оценка	Вес	Фамилия 1	Фамилия 2	Фамилия 3
Презентация	09.11.2022	10	16	0	8	8
Постановка задачи	09.11.2022	10	18	9	0	9
Определение структуры БД	22.11.2022	10	27	9	9	9
Загрузка данных	22.11.2022	10	18	3	5	10
Интерфейсы	01.12.2022	10	20	10	0	10
Справочники, отчеты	08.12.2022	10	30	10	10	10
Реализация подмножества	08.12.2022	10	30	10	10	10
Реализация подмножества с репликацией	15.12.2022	10	30	10	10	10
Отчет	15.12.2022	10	27	9	9	9
Итого		10	206	70	61	95

Для каждой бригады создается отдельная папка на Google-диске, к которой имеют доступ только преподаватель и члены этой бригады. В папку выкладываются все материалы, имеющие отношение к разработке. Также все

студенты имеют доступ к общей папке группы, в которой могут выкладываться методические материалы, примеры и т. п. Через документ «Вопросы и ответы» будет осуществляться общение с преподавателем.

Для контроля хода выполнения работ вся работа разделяется на этапы в форме заданий. Примерное содержание заданий приводится в следующем разделе. Бригада может уточнять содержание заданий и устанавливать сроки выполнения. Сдача заданий производится на практических занятиях. За несоблюдение сроков оценка за задание снижается.

4.5. Примерный перечень заданий

Задание 1.

Разместить в папку проекта предварительное описание проекта, в котором должны быть:

– постановка задачи с обоснованием необходимости распределенной БД;
– описание ролей пользователи системы;
– функциональные требования к системе, включая перечень функций для каждого роли пользователя (желательно диаграмма USE-CASE);
– описание данных (в виде ER-диаграммы) и структур таблиц;
– схема фрагментации, перечень используемых методов поддержки распределенности;
– описание выбранной платформы для реализации;
– определение реализуемого подмножества.

Задание 2.

1. Подготовить данные для заполнения БД в одном из форматов (для реализуемого подмножества):
   – SQL (предпочтительнее INSERT INTO …)
   – CSV
   – XML
   – JSON
2. Создать БД и загрузить в нее данные.
3. Подготовить несколько запросов SELECT для этих данных (2-3 запроса на участника).
4. Настроить экземпляры БД для репликации.

Данные можно подготовить в любой СУБД (MySQL, LibreOffice Base, Access и т. п.), а потом экспортировать в файл определенного формата.

В таблицах должно быть по несколько десятков записей.

Задание 3.

1. Подробно описать реализуемое подмножество (желательно использовать UML-диаграммы).

2. Спроектировать интерфейсы для пользователей.
3. Описать сценарии работы для каждой роли пользователя.
4. Реализовать простое приложение для редактирования таблицы.

Задание 4.

1. Подготовить отчетные формы по разработанным командам SELECT.
2. Подготовить данные на 2-х узлах распределенной системы, подлежащие репликации.
3. Описать минимальное подмножество проекта для реализации, включающее:

– функции администратора по редактированию таблиц;
– функции пользователя, затрагивающие репликацию;
– функции формирования отчетов, демонстрирующих результаты репликаций.

Задание 5.

Подготовить отчет по проекту (структура отчета в Приложении).

В кратком отчете описать реализацию подмножества проекта:

– введение (описание задачи из проекта);
– язык и среда программирования, инструменты, библиотеки;
– структура проекта (перечень файлов, назначение);
– диаграмма развертывания (UML) [не обязательно]);
– основные скриншоты;
– результаты тестирования (какие данные были введены, что получено в результате).

Задание 6.

1. Реализовать подмножество системы с репликацией.
2. Подготовить демонстрацию реализованного подмножества.
3. В кратком отчете описать, как было проверено действие репликации:

– способ размещения клиента и сервера;
– способы соединения;
– настройка MySQL (команды и строки в конфигурации);
– тестирование.

4.6. Критерии оценивания

Для контроля хода выполнения работ бригадир должен вести журнал работ, в котором для каждого этапа указывается сложность (вес) в баллах и срок выполнения. Этапы работ выбираются самостоятельно в соответствии с пунктами типовых заданий. Вес работы распределяется между участниками

бригады в зависимости от степени их участия. Преподаватель оценивает каждый этап по 10-балльной оценке, учитывая качество и своевременность выполнения. Итоговые баллы по участникам с учетом оценок нормируются в 10-балльную оценку по формуле

$$R = \frac{\sum_{i=1}^{n} b_i * c_i}{\sum_{i=1}^{n} c_i},$$

где b_i – оценка, полученная на выполнения i-го этапа, а c_i – вес этапа.

Библиографический список

1. Карпова И.П. Базы данных. Курс лекций и материалы для практических занятий: Учеб. пособие. – СПб., "Питер", 2013. – 240 с.
2. Коннолли Т., Бегг К. Базы данных: проектирование, реализация, сопровождение. Теория и практика. – 3-е изд.: Пер. с англ.: Уч. пос. – М.: Изд. дом "Вильямс", 2017. – 1439 с.
3. Проектирование реляционных баз данных: Метод. указания по выполнению домашнего задания по курсу "Базы данных" / МИЭМ НИУ ВШЭ; Сост.: Карпова И.П. – М., 2020. – 33 с.
4. Что такое подсистема Windows для Linux [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/windows/wsl/about (дата обращения 30.09.2023).
5. Установка Linux в Windows с помощью WSL [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/windows/wsl/install-win10 (дата обращения 30.09.2023).
6. MySQL 8.0 Reference Manual[Электронный ресурс]. URL: https://dev.mysql.com/doc/refman/8.0/en/binary-log.html (дата обращения 30.09.2023).
7. АСУ ТП. ЕСПД [Электронный ресурс]. URL: http://asutpseta.narod.ru/espd/espd_0.htm (дата обращения 30.09.2023).
8. Секреты удачного проектирования ИС (информационной системы) на примере строительства больницы [Электронный ресурс]. URL: https://habr.com/ru/articles/416525/ (дата обращения 31.08.2023).
9. Формирование требований и классификация требований [Электронный ресурс]. URL: https://analytics.infozone.pro/formation-requirements-and-classification-requirements (дата обращения 31.08.2023).
10. Модели жизненного цикла, принципы и методологии разработки программного обеспечения (ПО) [Электронный ресурс]. URL: https://evergreens.com.ua/ru/articles/software-development-metodologies.html (дата обращения 31.08.2023).

Полезные ссылки

1. Установка Docker в WSL: https://www.8host.com/blog/ustanovka-i-ispolzovanie-docker-v-ubuntu-20-04/.
2. Работа с docker через VisualStudio code: https://docs.microsoft.com/ru-ru/windows/wsl/tutorials/wsl-containers.
3. XML-редактор: https://freesoft.ru/windows/xml-notepad-2007
4. Генератор данных: http://www.generatedata.com/#generator
5. Диаграмма активностей: https://flexberry.github.io/ru/fd_activity-diagram.html
6. Визуализация данных при помощи DataTables.js: https://webdesign.tutsplus.com/ru/data-visualization-with-datatablesjs-and-highchartsjs--cms-29691t
7. Настройка плагина Datatables: http://www.itmathrepetitor.ru/javascript-nastroyka-plagina-datatable/
8. Django: https://www.djangoproject.com/
9. PostgreSWL: https://www.postgresql.org/
10. StarUML: https://staruml.io/download/

ПРИЛОЖЕНИЕ А. Пример проектирования РБД

Предметная область должна удовлетворять следующим требованиям:

- Иметь распределенный характер источников и/или потребителей данных, т.к. необходимо обосновать потребность в создании именно распределенной БД.
- Предполагать (по возможности) использование различных методов поддержки РБД: фрагментацию и репликацию, распределенные запросы и транзакции.

Примечание: если РБД поддерживается одним методом (например, только репликация), это влечет снижение итоговой оценки за работу.

Инфологическое проектирование

Анализ предметной области

Распределенная база данных создаётся для функционирования сети библиотек. РБД должна содержать всю необходимую информацию о книгах, филиалах и читателях, также необходима информация о наличии определенных книг в филиалах и об учете выдачи книг каждому читателю.

В соответствии с предметной областью система строится с учётом следующих требований:

- Есть несколько библиотек, одна из которых является главной, остальные — филиалами. Читателей обслуживают все они, но закупка книг и их распределение по филиалам осуществляется в центральной библиотеке.
- Есть общий список книг. Но в каждой библиотеке имеется свой список экземпляров книг, которые хранятся в данном филиале.
- Есть информация об издательствах. Одно издательство может издать несколько книг, каждая книга издается одним издательством.
- Авторы пишут произведения, а произведения публикуются в книгах. БД должна содержать информацию о произведениях в книгах.
- У одного произведения может быть несколько авторов, у автора может быть несколько произведений.
- Читатель может прийти в любой филиал или в центральную библиотеку и получить там читательский билет. По нему ему будут выдавать книги на дом, но только в том филиале, в котором он был зарегистрирован. В другие филиалы он может прийти со своим билетом и поработать в читальном зале этой библиотеки.
- Межбиблиотечный абонемент: если в текущем филиале нет нужной книги, читатель можете заказать ее из другого филиала. Ее привезут и сообщат ему об этом письмом по электронной почте. Читатель заказывает кни-

гу, этот заказ добавляется в систему. После оформления заказа эти данные пересылаются в центральную библиотеку. Далее центральная библиотека определяет, из какого филиала в какой будут перевозиться книги по всем заказам, вносит эту информацию в базу данных и отправляет эти данные по филиалам. Когда в филиал приходит машина с книгами, сотрудник филиала ставит заказу статус «выполнен», и триггер отсылает письмо заказчику, информируя того о прибытии книги.

Для создания ER-модели необходимо выделить сущности предметной области:

1) **Книги**. Атрибуты: id книги; название книги; год издания; ISBN.
2) **Авторы**. Атрибуты: id автора; ФИО; дата рождения; дата смерти.
3) **Издательства**. Атрибуты: id издательства; название издательства; адрес; телефон; почта.
4) **Библиотеки**. Атрибуты: id библиотеки; название библиотеки; адрес; дата открытия; дата закрытия.
5) **Произведения**. Атрибуты: id произведения; название произведения.
6) **Читатели**. Атрибуты: id читателя; ФИО; паспортные данные; адрес; телефон; email.
7) **Жанры.** Атрибуты: id жанра; название жанра (фантастика, детектив и т.д.).
8) **Типы.** Атрибуты: id типа; название типа (роман, рассказ и др.).

ER–диаграмма приведена на Рис. А.1.

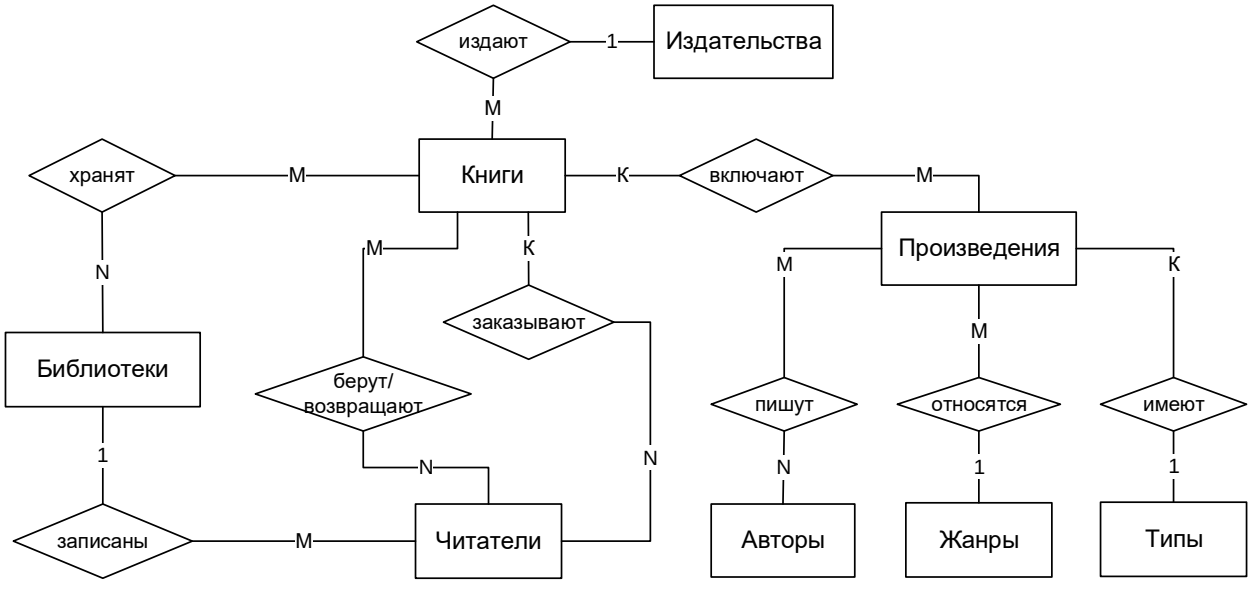

Рис. А.1 – ER–диаграмма ПрО «Сеть библиотек»

Анализ информационных задач и круга пользователей системы

Определим группы пользователей, их основные задачи и запросы к БД:
Библиотекари в филиалах:

- Регистрация новых читателей в филиале и выдача читательского билета.

- Получение информации о наличии книги или произведения, по запросу читателя.
- Учет выдачи/возврата книги, по запросу читателя.
- Регистрация заказа на доставку отсутствующей книги, которую запросил читатель.

Библиотекари в главном филиале (все те же запросы, что и для библиотекарей в филиалах, с добавлением специальных):

- Внесение информации о новых поступивших книгах и произведениях.
- Распределение прибывших книг по филиалам.
- Определение филиалов, из которых будет осуществляться перевозка книг (которые были заказаны читателями по МБА в филиалах, где отсутствовали данные книги).

Перевозчики:

- Получение информаций о том, из какого филиала необходимо перевести книги и в какой.

Директор:

- Чтение всех данных.
- Добавление и редактирование данных о библиотеках.

Заместитель директора (главный библиотекарь):

- Чтение всех данных.
- Заполнение и изменение справочных таблиц.
- Редактирование данных о библиотеках.

Читатель:

- Поиск книг по автору, названию, произведению, издательству, жанру, типу (по локальным таблицам в том филиале, где находится читатель).
- Удаленный запрос в центральную библиотеку, чтобы выяснить, в каких филиалах книга есть, если в текущем такой нет.
- Распределенный запрос к филиалам, чтобы уточнить, что книга действительно находится там (не перевезена в другой филиал).

Логическое проектирование реляционной БД

Преобразование ER–диаграммы в схему базы данных

Связи типа n:m реализуются через вспомогательные отношения (таблицы), которые содержат комбинации первичных ключей соответствующих исходных отношений. Связи типа 1:n реализуются с помощью внешнего ключа, который находится в подчиненном отношении. Полученная схема реляционной базы данных (РБД) приведена на Рис. А.2.

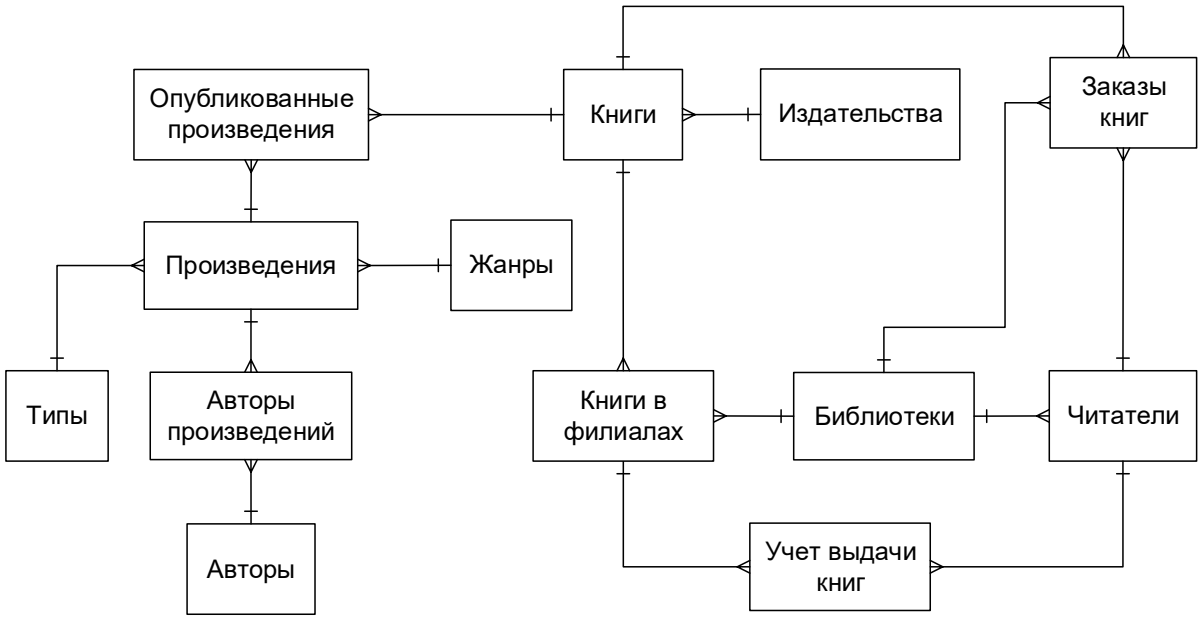

Рис. А.2. Схема РБД, полученная из ER–диаграммы «Сеть библиотек»

Составление реляционных отношений

Для тех отношений, которые не имеют уникальных полей, вводится суррогатный первичный ключ (атрибут *ID*).

Таблица А.1. Схема отношения *ЖАНРЫ* (Genres)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID	G_ID	N	3	Первичный ключ
Название жанра	G_NAME	V	50	Обязательное поле

Таблица А.2. Схема отношения *ТИПЫ* (Types)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID	T_ID	N	3	Первичный ключ
Название типы	T_NAME	V	50	Обязательное поле

Таблица А.3. Схема отношения *АВТОРЫ* (Authors)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID автора	A_ID	N	6	Первичный ключ
ФИО	A_NAME	V	100	Обязательное поле
Дата рождения	A_BORN	D		Обязательное поле
Дата смерти	A_DIED	D		

Таблица А.4. Схема отношения *ИЗДАТЕЛЬСТВА* (Publishers)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID издательства	P_ID	N	5	Первичный ключ
Название издатель-ства	P_NAME	V	100	Обязательное поле
Адрес	P_ADDRESS	V	100	Обязательное поле
Телефон	P_PHONE	V	30	Обязательное поле
E-mail	P_MAIL	V	30	Обязательное поле

Таблица А.5. Схема отношения *КНИГИ* (Books)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
Название книги	B_NAME	V	100	Обязательное поле
ID издательства	B_ID_PRESS	N	5	Обязательное поле, внешний ключ на таблицу *Издательства*
Год издания	B_PUBLISH_YEAR	N	4	Обязательное поле
ISBN	B_ISBN	V	25	Первичный ключ

Таблица А.6. Схема отношения *БИБЛИОТЕКИ* (Libraries)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID библиотеки	L_ID	N	3	Первичный ключ
Название библиотеки	L_NAME	V	100	Обязательное поле
Адрес	L_ADDRESS	V	100	Обязательное поле
Дата открытия	L_BEGIN	D		Обязательное поле
Дата закрытия	L_END	D		

Таблица А.7. Схема отношения *ЧИТАТЕЛИ* (Readers)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID читателя	R_ID	N	6	Первичный ключ
Фамилия	R_LNAME	V	60	Обязательное поле
Имя, отчество	R_FNAME	V	60	Обязательное поле
Паспорт	R_PASP	C	10	Обязательное поле
Адрес	R_ADDRESS	V	100	Обязательное поле
Телефон	R_PHONE	V	30	Обязательное поле
E-mail	R_MAIL	V	30	
ID филиала	R_ID_LIB	N	3	Обязательное поле, внешний ключ на таблицу *Библиотеки*

Таблица А.8. Схема отношения *ПРОИЗВЕДЕНИЯ* (Novels)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID произведения	N_ID	N	6	Первичный ключ
Название произведения	N_NAME	V	100	Обязательное поле
ID типа	N_ID_TYPE	N	3	Обязательное поле, внешний ключ на таблицу *Типы*
ID жанра	N_ID_GENRE	N	3	Обязательное поле, внешний ключ на таблицу *Жанры*

Таблица А.9. Схема отношения *ОПУБЛИКОВАННЫЕ ПРОИЗВЕДЕНИЯ* (Book_Novel)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*	
Номер п/п	BN_NUMBER	N	2	порядковый номер произведения в книге	
ID произведения	BN_ID_COMP	N	6	внешний ключ на таблицу Произведения	Составной первичный ключ
ID книги	BN_ID_BOOK	N	5	внешний ключ на таблицу Книги	

Таблица А.10. Схема отношения *АВТОРЫ ПРОИЗВЕДЕНИЙ* (Author_novel)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*	
ID произведения	AC_ID_COMP	N	6	внешний ключ на таблицу Произведения	Составной первичный ключ
ID автора	AC_ID_AUTH	N	6	внешний ключ на таблицу Авторы	

Таблица А.11. Схема отношения *КНИГИ В ФИЛИАЛАХ* (Book_library)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID	BL_ID	N	8	Первичный ключ
ID библиотеки	BL_ID_LIBRARY	N	3	Обязательное поле, внешний ключ на таблицу Библиотеки
ID книги	BL_ID_BOOK	V	25	Обязательное поле, внешний ключ на таблицу Книги
Контрольный экземпляр	BL_CONTROL	C	1	1/0 – да/нет
Забронирован	BL_BOOKING	DT		дата-время бронирования
ID бронировавшего читателя	BL_ID_READER	N	6	Необязательное поле, внешний ключ на таблицу Читатели

Таблица А.12. Схема отношения *ЗАКАЗЫ КНИГ* (Book_orders)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID	OL_ID	N	6	Первичный ключ
ID книги	OL_BOOK	V	25	Обязательное поле, внешний ключ на таблицу *Книги*
ID читателя	OL_READER	N	6	Обязательное поле, внешний ключ на таблицу *Читатели*
Дата формирования заказа	OL_CREATE_DATA	D		Обязательное поле
Дата выполнения заказа	OL_EXEC_DATA	D		
ID филиала отправления	OL_ID_LIB	N	6	внешний ключ на таблицу *Библиотеки*
Статус	OL_STATUS	C	10	Значения – 'принят', 'отправлен', 'доставлен', 'отказано'. Обязательное поле.

Таблица А.13. Схема отношения *УЧЕТ ВЫДАЧИ КНИГ* (Abonement)

Содержимое поля	*Название*	*Тип*	*Длина*	*Примечание*
ID	V_ID	N	6	Первичный ключ
ID читателя	V_ID_READER	N	6	Обязательное поле, внешний ключ на таблицу *Читатели*
ID книги в филиалах	V_ID_BOOK_IN_LIB	N	8	Обязательное поле, внешний ключ на таблицу *Книги в филиалах*
Дата выдачи	V_DATE_EXT	D		Обязательное поле
Дата возврата	V_DATE_RET	D		
Статус	V_STATUS	V	10	Значения – 'выдана' и 'утеряна', по умолчанию – 'выдана' (*)
Сумма штрафа за порчу/утерю	V_FINE	N	5	

*** Примечание:** если книга возвращена, то значение поля Статус – null. Такой подход позволит создать по этому полю эффективный индекс, который будет содержать только данные об утерянных и выданных книгах, а большинство строк в этой таблице будут иметь неопределенное значение данного поля, которое не входит в индекс. Такой индекс будет занимать мало места и позволит быстро находить выданные или утерянные книги.

Примечание: если полученные в ходе проектирования БД таблицы не удовлетворяют 3НФ, необходимо выполнить этап нормализации отношений.

Определение дополнительных ограничений целостности

1. В поле *Сумма штрафа за потерю/порчу* хранится сумма штрафа меньше 10000 руб. Значение поля по умолчанию – 0.
2. Один и тот же экземпляр книги не может быть одновременно выдан нескольким людям.

Описание групп пользователей и прав доступа

Опишем для каждой группы пользователей права доступа к каждой таблице. Права доступа должны быть распределены так, чтобы для каждого объекта БД был хотя бы один пользователь, который имеет право добавлять и удалять данные из объекта.

Таблица А.14. Права доступа к таблицам для групп пользователей

Таблицы	Группы пользователей (роли)				
	Библиотекари филиалов	Библиотекари главного филиала	Перевозчики	Директор	Заместитель директора
Жанры	S	S		S	SUID
Типы	S	S		S	SUID
Издательства	S	SUID		S	S
Книги	S	SUID	S	S	S
Авторы	S	SUID			
Библиотеки	S	S	S	SUI	SI
Читатели	SUID*	SUID*		S	S
Произведения	S	SUID		S	S
Книги в филиалах	SUID*	SUID*	S	S	S
Учет выдачи книг	SUID*	SUID*		S	S
Заказы книг	SUID*	SUID	SU	S	S
Опубликованные произведения	S	SUID		S	S
Авторы произведений	S	SUID		S	S

* – права UID только для своего филиала

Составление схемы фрагментации

Для создания схемы фрагментации необходимо определить структуру узлов сети. Будем считать, что есть узел в центральной библиотеке и несколько филиалов, в каждом из которых есть свой локальный сервер БД (Рис. А.3). Сеть может быть полносвязной, при этом необязательно, чтобы каждый узел был связан со всеми другими узлами непосредственно.

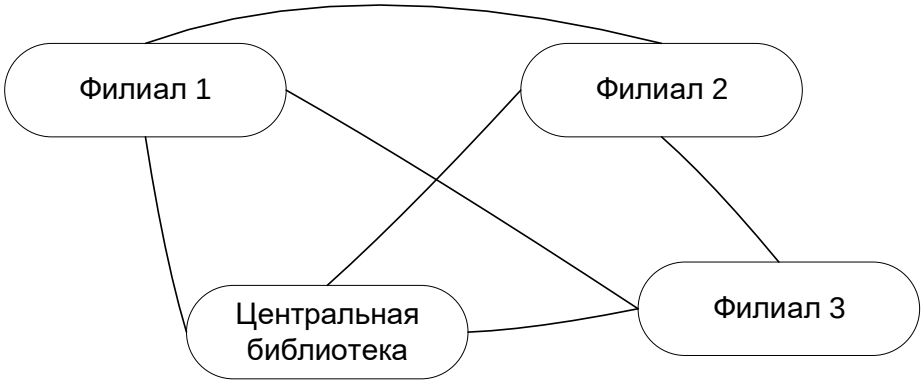

Рис. А.3. Схема узлов сети

Для создания схемы фрагментации необходимо определить методы поддержки распределенных данных. Закупка книг осуществляется в центральной библиотеке, поэтому в случае использования репликации основная копия хранится там, у филиалов – реплики (при консолидации данных в центральной библиотеке хранится консолидированная копия). Схема фрагментации приведена на Рис. А.4.

1) **Справочные таблицы.** К ним относятся **Жанры, Типы, Издательства, Библиотеки**. Эти таблицы формируются централизованно, они имеют относительно небольшой размер и меняются редко. Для таких таблиц наиболее подходящей является **репликация с основной копией на основе моментальных снимков, полная регенерация раз в сутки**.

2) Основные таблицы **Авторы, Книги, Произведения, Авторы произведений, Опубликованные произведения**. Они тоже формируются централизованно, но меняются чаще, чем справочные таблицы, и имеют гораздо больший размер (сотни тысяч или миллионы записей). Для этих таблиц наиболее подходящей является **репликация с основной копией на основе моментальных снимков, полная регенерация раз в неделю, инкрементная регенерация раз в сутки**.

3) Основная таблица **Читатели**. Данные этой таблицы могут меняться достаточно часто, причем нужно обеспечить возможность записывать новых читателей в любой филиал и получать данные читателей из других филиалов, чтобы выдавать книги в читальном зале. Здесь наиболее подходящей является **репликация без основной копии с асинхронным распространением изменений**. При асинхронном распространении изменений возможны кон-

фликты добавления, изменения и удаления записей, поэтому надо предусмотреть способы их разрешения.

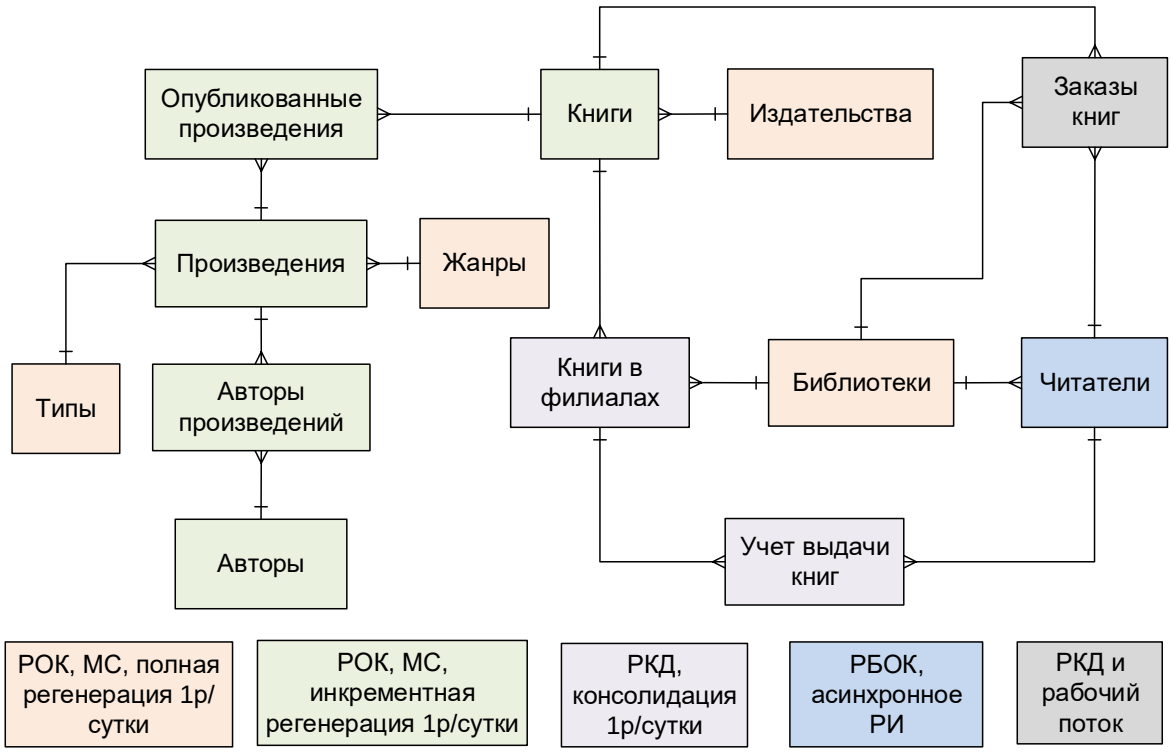

Рис. А.4. Схема фрагментации РБД

Обозначения на Рис. А.4:

РОК – репликация с основной копией
РБОК – репликация без основной копии
РКД – репликация с консолидацией данных
МС – моментальные снимки
РИ – распространение изменений

4) **Книги в филиалах: репликация с основной копией на основе консолидации данных**. Данные в эту таблицу добавляются филиалами. В каждом филиале – своя локальная таблица, данные которой хранятся в виде консолидированной копии в центральной библиотеке (регенерация раз в сутки).

5) **Учет выдачи книг: также репликация с консолидации данных**. В данную таблицу заносятся данные по книгам, которые взял читатель. Консолидация данных происходит в центральной библиотеке, данные в центральную библиотеку передаются раз в сутки. Консолидированная копия играет роль резервной копии и повышает надежность БД.

6) **Заказы книг: репликация с консолидации данных и рабочим потоком**. Данные о заказе вносятся библиотекарем в локальную таблицу филиала. При добавлении нового заказа указывается книга и читатель, а также проставляется статус, что заказ принят. Затем с помощью автоматизированной процедуры данные отправляются в консолидированную копию (в центральную библиотеку, ЦБ), заказ помечается как поступивший на обработку, и

филиал теряет право изменять эти данные, а центральная библиотека приобретает это право (это суть рабочего потока). Далее сотрудник центральной библиотеки определяет, из какого филиала в какой будут перевозиться книги по всем заказам, и вносит эти данные в общую таблицу в центральной библиотеке. Когда заказы отправляются по местам назначения, сотрудник в таблице (в ЦБ) меняет статус заказа и центральная библиотека теряет право изменять эти данные. Они с помощью триггера отправляются в локальную таблицу филиала, который теперь может их менять. По прибытии книг в филиал сотрудник меняет статус полученного заказа, и триггер отправляет сообщение читателю с информацией о прибытии заказа.

Кроме репликации предполагается использование **удаленных запросов**: если в конкретном филиале нет в наличии определенной книги, можно отправить удаленный запрос на центральный узел для того, чтобы сообщить читателю, в каких филиалах эта книга есть, чтобы он мог туда поехать (в читальный зал, в котором всегда можно взять контрольный экземпляр книги: он не выдается на руки). Если запрос возвращает ошибку из-за сбоя в работе центрального узла, то система должна выполнить **распределенный запрос** по всем узлам, кроме центрального. Если и этот запрос возвращает ошибку, то система должна выполнить **набор отдельных удаленных запросов** к каждому узлу, игнорируя ошибки, и возвращать данные от тех узлов, которые ответили на запрос.

При подтверждении наличия книги в определенном филиале читатель может забронировать ее для себя на сутки. Бронирование выполняется с помощью **удаленной транзакции** и заключается в том, что в локальной таблице Книги_в_филиалах того узла, в котором будет забронирована книга, для этой книги устанавливается текущие дата/время начали брони и идентификатор забронировавшего ее читателя. Если там уже установлена другая активная бронь, то транзакция не может быть выполнена (должна вернуть ошибку). Если предыдущая бронь закончилась (прошло более суток в момента ее установки), то предыдущая бронь заменяется новой.

Раз в сутки (по расписанию) необходимо запускать процедуру, снимающую закончившуюся бронь с книг.

ПРИЛОЖЕНИЕ Б. Руководство по созданию репликации в системе Windows без использования средств виртуализации

Пересадов Владислав

Данное руководство описывает процесс выполнения master-slave репликации на Windows с использованием двух серверов СУБД MySQL разных версий.

1. Если вы это читаете, то, скорее всего, у вас уже есть ER-диаграмма, схема БД, возможно даже нормализованная.
2. Для выполнения работы необходимо поставить MySQL Installer: https://dev.mysql.com/downloads/installer/ .
3. Далее в установленном приложении необходимо нажать “Add …”, выбрать минимум 2 mysql сервера разных версий, как, например, изображено на Рис. Б.1. Также желательно установить workbench, т.к. пример будет показываться на нем.

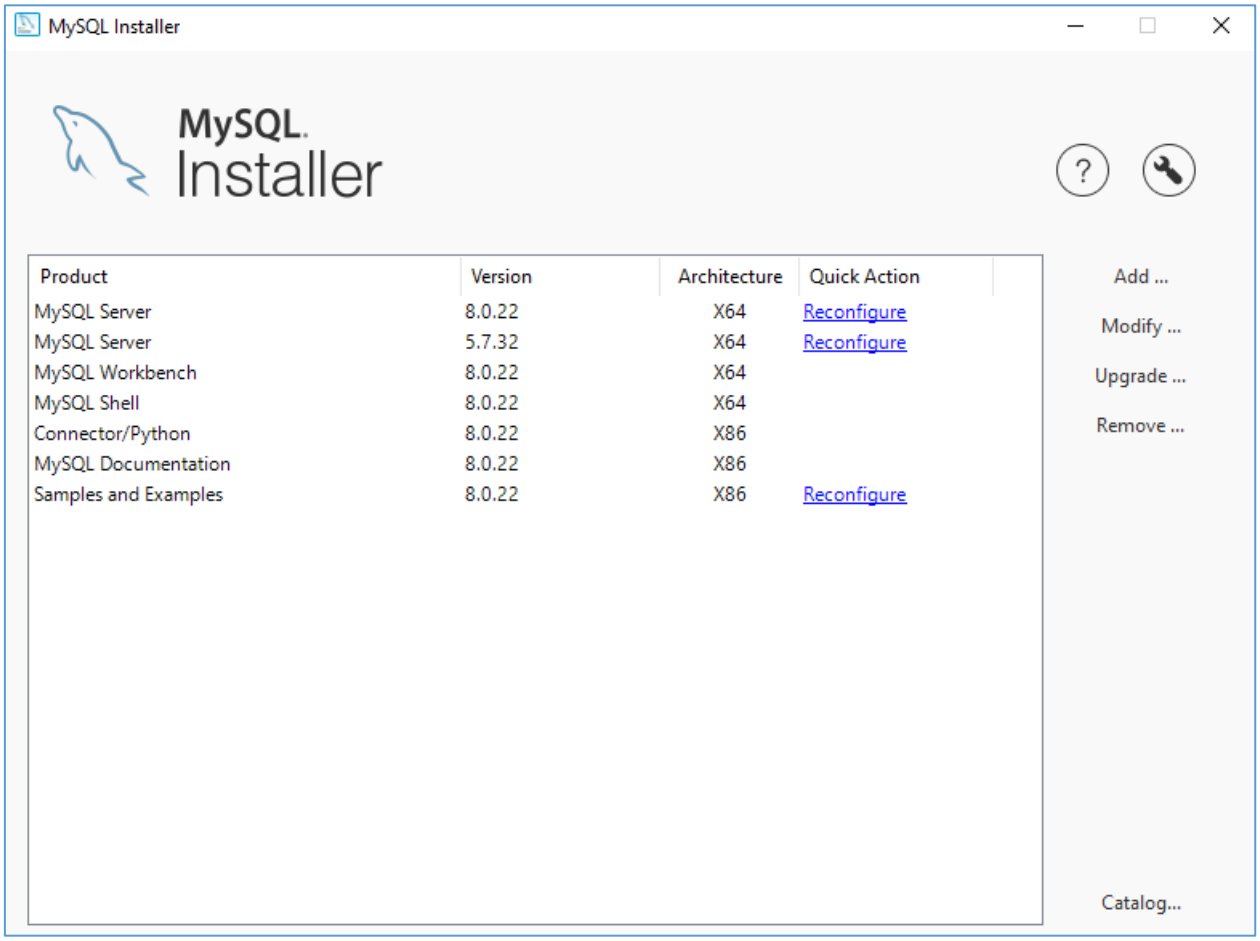

Рис. Б.1. Выбор серверов для репликации

Важно: серверы должны иметь либо идентичную версию, либо master-slave-структура должна строиться с учетом обратной совместимости, т.е. master – это всегда самый старший по версии сервер, а slave – младший. (На Рис. Б.1 сервер 5.7 будет мастером, а 8.0 - подчиненным).

4. При установке серверов необходимо будет их сконфигурировать – назначить каждому свой уникальный порт; (далее в примере используются порт 3306 для master-сервера, и 3307 – для slave-сервера; остальные надстройки можно оставить по умолчанию).

После шага 4 у вас уже: установлены два mysql-сервера и workbench (wb).

5. Соединяемся через wb с master-сервером: необходимо ввести стандартный внутренний ip-адрес и порт, который указывали во время установки сервера на прошлом шаге (Рис. Б.2).

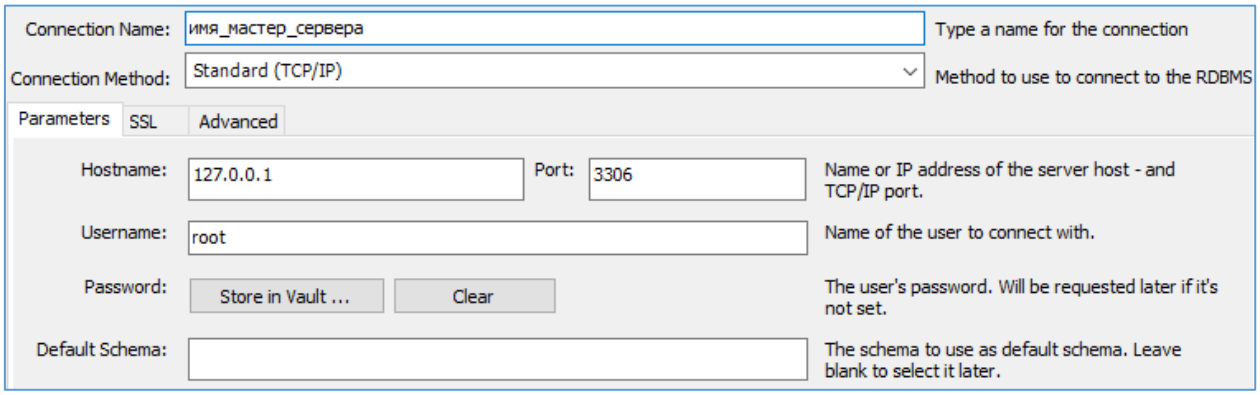

Рис. Б.2. Ввод данных

При проведении «Test connection» должно быть показано сообщение об успешном установлении соединения (Рис. Б.3).

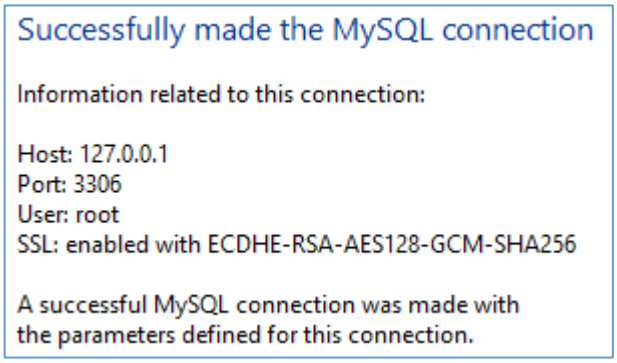

Рис. Б.3. Сообщение о подключении

Примечание: если вместо Рис. Б.3 появилось сообщение об ошибке, проверьте, запустился ли сервер (диспетчер задач -> вкладка «Службы» -> ищем «MySQL??» где ?? – номер версии сервера (если не меняли имя службы сервера при установке)). Служба имеет статус «Отключено», то ее можно включить оттуда же. Если сервер почему-то не запускается через диспетчер задач, то идем в приложение MySQL Installer, там возле серверов, в разделе “Quick Action” будет действие “Reconfigure” (Рис. Б.4), нажимаем на него. Далее жмем Next-ы, смотрим что порт выставлен правильно, вводим пароль, нажимаем Check, снова жмем Next-ы и Execute. Проверяем в диспетчере, запустился ли сервер. Если не помогло – ищем пути решения проблемы с помощью интернета.

Product	Version	Architecture	Quick Action
MySQL Server	8.0.22	X64	Reconfigure
MySQL Server	5.7.32	X64	Reconfigure
MySQL Workbench	8.0.22	X64	
MySQL Shell	8.0.22	X64	
Connector/Python	8.0.22	X86	
MySQL Documentation	8.0.22	X86	
Samples and Examples	8.0.22	X86	Reconfigure

Рис. Б.4. Повторный выбор сервера

6. То же самое делаем для slave-сервера.
7. Создаем базу данных по имеющейся структуре на master-сервере, заполняем ее данными, пишем триггеры, пишем запросы – в общем, работаем как с обычной БД.
8. Теперь необходимо следовать этому руководству: https://www.youtube.com/watch?v=APAmsHAYUiI .

Примечание: файлы для изменения конфигурации серверов находятся по адресам C:\ProgramData\MySQL, в папках соответствующих серверов (прим.: C:\ProgramData\MySQL\MySQL Server 5.7; C:\ProgramData\MySQL\MySQL Server 8.0)

ПРИЛОЖЕНИЕ В. Структура итогового отчета

Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики

Департамент компьютерной инженерии

Отчёт по дисциплине
«Распределенные базы данных и сетевые вычисления»

«ПРОЕКТИРОВАНИЕ РАСПРЕДЕЛЕННОЙ БАЗЫ ДАННЫХ "....."»

ВЫПОЛНИЛИ:

Фамилия И.О., группа ...

…...

Москва 202_

Содержание

ПРИЛОЖЕНИЕ Г. Список источников для настройки репликации в СУБД PostgreSQL

Репликация в PostgreSQL:

1. Документация PostgreSQL, раздел 20.6. "Репликация". https://www.postgresql.org/docs/current/runtime-config-replication.html
2. Андрей Бирюков. Репликации в PostgreSQL. Блог компании OTUS. https://habr.com/ru/companies/otus/articles/710956/
3. Sysadminium. База знаний системного администратора. https://sysadminium.ru/replikaciya_v_postgresql/
4. Академия Selectel. Как настроить репликацию в PostgreSQL. https://selectel.ru/blog/tutorials/how-to-set-up-replication-in-postgresql/?ysclid=loveycd0cl41993043
5. Потоковая репликация в PostgreSQL – короткое введение. https://prudnitskiy.pro/post/2018-01-05-pgsql-replica/