**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение**

**Высшего профессионального образования**

**Московский технический университет связи и информатики**

Факультет повышения квалификации

**Лабораторная работа №1**

**по дисциплине**

**«Интеллектуальные системы»**

Выполнил:

магистрант гр. 3мпп1901

Каргальцев М. М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись дата

Проверил:

к.т.н., доцент каф. МКиИТ

Махров С.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись дата

**Москва 2020**

# Цель работы:

Целью работы является изучение реляционных баз данных и типовых запросов CRUD.

# Задание:

1. Развернуть PostgreSQL в Docker-контейнере;
2. Установить DBeaver;
3. Придумать предметную область;
4. Создать схему для своей предметной области в базе данных;
5. В созданной схеме создать таблицы (не менее 7 таблиц), описывающие реализующие выбранную предметную область. Таблицы и связи должны быть нормализованы до нормальной формы Бойса-Кодда (НФБК). В тексте отчета сделать ссылку на Приложение 1, в котором указать код DDL запросов;
6. Наполнить текстовыми данными таблицы. В тексте отчета сделать ссылку на Приложение 2, в котором указать код DML запросов;
7. Написать Dockerfile, который развертывает виртуальный контейнер с PostgreSQL и создает схему вместе со всеми таблицами и данными в них. Код Dockerfile вместе представить в Приложении 3.
8. Извлечь готовый Docker образ и сохранить локально.
9. Выполнить тестовые запросы CRUD:

* различные запросы на INSERT, содержащие данные к таблицам;
* 3 запроса на выборку данных, каждый запрос должен иметь блок WHERE, в котором должны быть связи с как минимумом 2-мя другими таблицами и какие-либо условия фильтрации данных;
* 1 простой запрос на UPDATE к любой таблице по критериям, указанным в блоке WHERE;
* 1 сложный запрос на UPDATE - обновление таблицы A в зависимости от данных в таблице B (придумать критерии);
* 1 простой запросы на DELETE - к любой таблице по критериям, указанным в блоке WHERE;
* 1 сложный запрос на DELETE - удаление записей из таблицы A в зависимости от данных в таблице B (придумать критерии);

1. Изучить и выполнить запрос SELECT FOR UPDATE.
2. Предоставить отчет на в slack в директ сообщение: <https://join.slack.com/t/intsystems/shared_invite/zt-dgvb13xu-VJ1itP7HN33PoJXcMLGb4Q>

к отчету также приложить Dockerfile и образ Docker-контейнера.

# Ход выполнения работы:

(Содержимое отчета:

1. Описание выбранной предметной области - текст на четверть страницы отчета;
2. ER-диаграмма с указанием типов данных и комментарием к каждому столбцу (к названиям столбцов можно давать комментарии прямо в таблице). В тексте отчета сослаться на Приложение 1 к отчету, в котором разместить DDL;
3. Описание того, какими тестовыми данными наполнили БД. В тексте отчета сослаться на Приложение 2 к отчету, в котором разместить DML;
4. Описание того, как создавали Dockerfile.
5. Описание того, как выгружали Docker-образ.
6. Тестовые запросы CRUD и результат выполнения каждого запроса.
7. Тестовый запрос SELECT FOR UPDATE и результат выполнения запроса.

)

# Ключевые вопросы:

1. Что такое SQL, DDL, DML, DCL, TCL?
2. Что такое CRUD?
3. Что такое ER диаграмма?
4. Зачем нужен оператор SELECT FOR UPDATE?
5. Зачем нужен оператор MERGE и есть ли он в PostgreSQL?
6. Какой уровень изоляции транзакций используется по умолчанию в PostgreSQL, какие его собенности и как его поменять?
7. Чем Dockerfile отличается от Docker-образа? Когда нужно писать Dockerfile, а когда выгружать образ?
8. В чем отличие Dockerfile от docker-compose.yml?

# Вывод:

В результате выполнения лабораторной работы, были изучены основы работы с ПО для автоматизации развёртывания и управления приложениями Docker. Появились практические навыки работы с СУБД Postgres, а также ознакомились с процессом развертывания Postgres в контейнере Docker.

Приложение 1

1. Таблица egais\_company

create table exc.egais\_company(

egais\_company\_id serial primary key -- id

,egais\_company\_name varchar(512) -- название организации

,egais\_company\_address varchar(1024) --адрес организации

,fsrar\_id varchar(64) UNIQUE --уникальны код организации

,is\_active bool DEFAULT false); --активность УТМ организации

2. Таблица egais\_class

create table exc.egais\_class(

egais\_class\_id serial primary key --id

,egais\_class\_name varchar(255) --название класса

,egais\_class\_description varchar(256)); --описание класса

3. Таблица egais\_status

create table exc.egais\_status(

egais\_status\_id serial primary key --id

,egais\_status\_name varchar(256) --название статуса

,egais\_status\_description varchar(256)); --описание

4. Таблица utm

create table exc.utm(

utm\_id serial primary key --id

,utm\_name varchar(256) --название УТМ

,utm\_ip cidr unique --ip адрес сервера

,is\_active bool DEFAULT false); --активность УТМ

5. Таблица egais\_type

create table exc.egais\_type(

egais\_type\_id serial primary key --id

,egais\_class\_id int REFERENCES exc.egais\_class (egais\_class\_id) NOT NULL

--id внешняя ссылка на egais\_class

,egais\_type\_name varchar(256) --название типа

,egais\_type\_description varchar(256)); --описание типа

6. Таблица egais\_data

create table exc.egais\_data(

egais\_data\_id serial primary key --id

,egais\_type\_id int REFERENCES exc.egais\_type (egais\_type\_id) NOT NULL -

--id ссылка на egais\_type

,egais\_content xml --данные передаваемые в УТМ

,egais\_status\_id int REFERENCES exc.egais\_status (egais\_status\_id) NOT NULL

--id ссылка на egais\_status

,direction int NOT NULL --направление движения (входящий/исходящий)

,url varchar(256) NOT NULL --полный url адрес документа

,utm\_id int REFERENCES exc.UTM (utm\_id) NOT NULL);

--id ссылка на utm

7. Таблица egais\_document

create table exc.egais\_document(

egais\_document\_id serial primary key --id

,egais\_document\_number varchar(64) NOT NULL –номер документа

,egais\_data\_id int REFERENCES exc.egais\_data (egais\_data\_id ) NOT NULL

--id ссылка на egais\_data

,egais\_company\_id int REFERENCES exc.egais\_company(egais\_company\_id ) NOT NULL

--id сылка на egais\_company

,create\_time timestamp); --время вставки записи

Приложение 2

1. insert into exc.utm(

utm\_ip

,utm\_name)

values

('192.168.6.149','retail')

,('192.168.6.154','wholesale')

2. insert into exc.egais\_class(

egais\_class\_name

,egais\_class\_description )

values

('waybill', 'despatch')

,('actwriteoff', 'return')

3. insert into exc.egais\_type (

egais\_class\_id

,egais\_type\_name

,egais\_type\_description )

values

(1, 'waybill\_v3', 'despatch\_v3')

,(2, 'actwriteoff\_v3', 'return\_v3')

4. insert into exc.egais\_company (

egais\_company\_name

,egais\_company\_address

,fsrar\_id

,is\_active )

values

('tander', 'moscow', '010000000580', true)

,('f5retail', 'moscow', '010000006337', false)

5. insert into exc.egais\_status (

egais\_status\_name

,egais\_status\_description )

values

('new', 'new')

,('sent', 'sent')

6. insert into exc.egais\_document (

egais\_document\_number

,egais\_company\_id

,create\_time )

values(

'001/20', 1, current\_timestamp)

,('002/20', 2, current\_timestamp)

7. insert into exc.egais\_data(

egais\_type\_id

,egais\_content

,egais\_status\_id

,direction

,url

,utm\_id )

values

(1, '<?xml version="1.0" encoding="UTF-8"?>', 1, 1, '/opt/in/1235', 1)

,(2, '<?xml version="1.0" encoding="UTF-8"?>', 2,-1, '/opt/out/12424', 2)

Приложение 3

FROM postgres

COPY /db/main.sql /docker-entrypoint-initdb.d

Отчет по лабораторной работе № 1

1. В качестве предметной области был выбран фрагмент базы данных сервиса обмена данных с универсальным транспортным модулем(УТМ) ЕГАИС(Единая Государственная Автоматизированная Информационная Система). Фрагмент описывает небольшую часть обмена данных с УТМ. Сервис представляет собой Windows Service на писанный на C# и использующий в качестве сервера MSSQLServer.

Таблица egais\_data(Приложение 1 п. 6) представляет собой хранилище исходящих и входящих данных, ссылающуюся на таблицу egais\_type(Приложение 1 п. 5) и определяющую тип данных. В свою очередь egais\_type ссылается на egais\_class(Приложение 1 п. 2) , которая классифицирует типы и версии документов ЕГАИС. Таблица utm(Приложение 1 п. 6) - это объекты серверов УТМ, с которых осуществляется отправка данных в ЕГАИС. Таблица egais\_company(Приложение 1 п. 1) содержит участников документооборота с ЕГАИС, каждая запись имеет уникальный код в системе(FSRAR\_ID). Все документы хранятся в таблице egais\_document (Приложение 1 п. 7). Таблицы egais\_status(Приложение 1 п. 3) описывает состояние документа в ЕГАИС.

2. ER — диаграмма egaisExchange

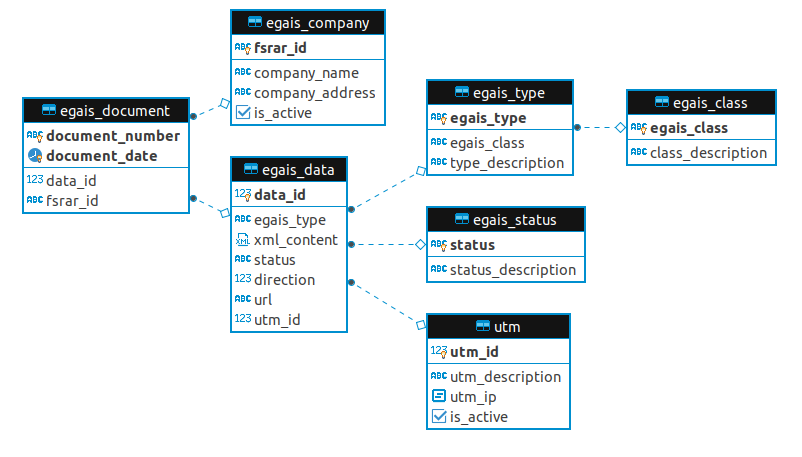


Рис. 1 - ER диаграмма egaisExchange

Типы данных и описание столбцов указаны в Приложении 1.

3. Для наполнения использовались следующие тестовые данные:

utm(Приложение 2 п.1) — два объекта утм(розница и опт)

egais\_class(Приложение 2 п.2) – два класса ЕГАИС(отгрузка и списание/возврат)

egais\_type(Приложение 2 п.3) – два типа документа(отгрузка версии 3 и возврат версии 3)

egais\_company(Приложение 2 п.4) – две организации

egais\_status(Приложение 2 п.5) – статусы Новый и отправлен

egais\_document(Приложение 2 п.6) – тестовые документы

egais\_data(Приложение 2 п.7) – исходящие данные.

4. Dockerfile(Приложение 3) состоит из команды COPY, которая копирует скрипты sql(db.sql, schema.sql, ddl.sql, dml.sql) из каталога db, который в свою очередь находится в каталоге egaisExchange и четырех команд CMD которые по очереди формируют базу данных с тестовыми данными.

5. Docker – образ сформирован следующей командой:

sudo docker save -o egaisexchangedb.tar postgres

6. CRUD запросы и результаты выполнения.

select et.egais\_type\_name, ed.url

from exc.egais\_data ed

join exc.egais\_type et

on et.egais\_type\_id = ed.egais\_type\_id

join exc.egais\_class cl

on cl.egais\_class\_id = et.egais\_class\_id

where ed.direction = 1

and cl.egais\_class\_id = 1

Результат выполнения:

|  |  |
| --- | --- |
| egais\_type\_name | url |
| waybill\_v3 | /opt/in/1235 |

select ed.egais\_document\_number , edt.url, ec.egais\_company\_name

from exc.egais\_document ed

join exc.egais\_data edt

on edt.egais\_data\_id = ed.egais\_data\_id

join exc.egais\_company ec

on ec.egais\_company\_id = ed.egais\_company\_id

where ed.egais\_company\_id = 1

Результат выполнения:

|  |  |  |
| --- | --- | --- |
| egais\_document\_number | url | egais\_company\_name |
| 001/20 | /opt/in/1235 | tander |

select ed.direction, es.egais\_status\_name, u.utm\_ip

from exc.egais\_data ed

join exc.egais\_status es

on es.egais\_status\_id = ed.egais\_status\_id

join exc.utm u

on u.utm\_id = ed.utm\_id

where u.utm\_id = 2

Результат выполнения:

|  |  |  |
| --- | --- | --- |
| direction | egais | egais\_status\_name |
| -1 | sent | 192.168.6.154/32 |

update exc.egais\_document

set egais\_company\_id = 2

where egais\_document\_number = '001/20'

Результат выполнения: 1 row(s) updated - 1ms

update exc.egais\_data

set utm\_id = 1

from exc.utm

where exc.egais\_data.utm\_id = exc.utm .utm\_id

and exc.utm .utm\_ip = '192.168.6.149/32'

Результат выполнения: 1 row(s) updated - 1ms

delete from exc.egais\_document where egais\_document\_id is null

Результат выполнения: 0 row(s) updated - 1ms

delete from exc.egais\_type USING exc.egais\_class

where exc.egais\_type.egais\_class\_id = exc.egais\_class.egais\_class\_id

and exc.egais\_type.egais\_type\_name is null

Результат выполнения: 0 row(s) updated – 1ms

7. select for update

select ed.egais\_document\_number, ed.egais\_data\_id

from exc.egais\_document ed

where ed.egais\_document\_number is not null

limit 2 for update

Результат выполнения:

|  |  |
| --- | --- |
| egais\_document\_number | egais\_data\_id |
| 002/20 | 1 |
| 001/20 | 2 |

Ответы на вопросы:

1. Что такое SQL, DDL, DML, DCL, TCL?

Structured Query Language (SQL) — язык структурированных запросов, с помощью него пишутся специальные запросы (SQL инструкции) к базе данных с целью получения этих данных из базы и для манипулирования этими данными.

Data Definition Language (DDL) – это группа операторов определения данных. Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы, т.е. создаем, изменяем и удаляем их.

В эту группу входят следующие операторы:

CREATE – используется для создания объектов базы данных;

ALTER – используется для изменения объектов базы данных;

DROP – используется для удаления объектов базы данных.

Data Manipulation Language (DML) – это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы, т.е. манипулировать ими.

В эту группу входят самые распространённые операторы языка SQL:

* SELECT – осуществляет выборку данных;
* INSERT – добавляет новые данные;
* UPDATE – изменяет существующие данные;
* DELETE – удаляет данные.

Data Control Language (DCL) – группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций над объектами базы данных.

Сюда входят:

GRANT – предоставляет пользователю или группе разрешения на определённые операции с объектом;

REVOKE – отзывает выданные разрешения;

DENY– задаёт запрет, имеющий приоритет над разрешением.

Transaction Control Language (TCL) – группа операторов для управления транзакциями. Транзакция – это команда или блок команд (инструкций), которые успешно завершаются как единое целое, при этом в базе данных все внесенные изменения фиксируются на постоянной основе или отменяются, т.е. все изменения, внесенные любой командой, входящей в транзакцию, будут отменены.

Сюда можно отнести:

BEGIN TRANSACTION – служит для определения начала транзакции;

COMMIT TRANSACTION – применяет транзакцию;

ROLLBACK TRANSACTION – откатывает все изменения, сделанные в контексте текущей транзакции;

SAVE TRANSACTION – устанавливает промежуточную точку сохранения внутри транзакции.

1. Что такое CRUD?

CRUD — аббревиатура[, обозначающая четыре базовые функции, используемые при работе с](https://ru.wikipedia.org/wiki/Акроним)[базами данных](https://ru.wikipedia.org/wiki/База_данных)[: создание (](https://ru.wikipedia.org/wiki/CRUD" \l "cite_note-1)[англ.](https://ru.wikipedia.org/wiki/Английский_язык) create), чтение (read), модификация (update), удаление (delete). В [SQL](https://ru.wikipedia.org/wiki/SQL) этим функциям операциям соответствуют операторы [Insert](https://ru.wikipedia.org/wiki/Insert_(SQL)) (создание записей), [Select](https://ru.wikipedia.org/wiki/Select_(SQL)) (чтение записей), [Update](https://ru.wikipedia.org/wiki/Update_(SQL)) (редактирование записей), [Delete](https://ru.wikipedia.org/wiki/Delete_(SQL)) (удаление записей).

1. Что такое ER диаграмма?

Диаграмма привязки сущности ER - это блок-схемы, которые иллюстрируют, как «сущности» (люди, объекты или концепции) относятся друг к другу в системе.

1. Зачем нужен оператор SELECT FOR UPDATE?

В режиме FOR UPDATE строки, выданные оператором SELECT, блокируются как для изменения. При этом они защищаются от блокировки, изменения и удаления другими транзакциями до завершения текущей. То есть другие транзакции, пытающиеся выполнить UPDATE, DELETE,

SELECT FOR UPDATE, SELECT FOR NO KEY UPDATE, SELECT FOR SHARE или SELECT FOR KEY SHARE с этими строками, будут заблокированы до завершения текущей транзакции; и наоборот, команда SELECT FOR UPDATE будет ожидать окончания параллельной транзакции, в которой выполнилась одна из этих команд с той же строкой, а затем установит блокировку и вернёт изменённую строку (или не вернёт, если она была удалена).

Режим блокировки FOR UPDATE также запрашивается на уровне строки любой командой DELETE и командой UPDATE, изменяющей значения определённых колонок.

1. Зачем нужен оператор MERGE и есть ли он в PostgreSQL?

Merge — оператор языка SQL, который позволяет слить данные одной таблицы с данными другой таблицы. При слиянии таблиц проверяется условие, и если оно истинно, то выполняется Update, а если нет - Insert. Причём нельзя изменять поля таблицы в секции Update, по которым идет связывание двух таблиц. Данные изменяются или добавляются только для таблицы в предложении MERGE INTO, таблица в предложении USING остается без изменений.

Есть поддержка этого оператора в PostgreSQL 8.4devel Documentation. Присутствует подробная информация, а так же параметры которые используются в данном операторы, получаемые вывод при использовании данного оператора.

1. Какой уровень изоляции транзакций используется по умолчанию в PostgreSQL, какие его особенности и как его поменять?

По умолчанию в PostgreSQL используется уровень изоляции Read Committed. Такой уровень изоляции всегда позволяет видеть изменения внесённые успешно завершёнными транзакциями в оставшихся параллельно открытых транзакциях. В транзакции, работающей на этом уровне, запрос SELECT (без предложения FOR UPDATE/SHARE) видит только те данные, которые были зафиксированы до начала запроса; он никогда не увидит незафиксированных данных или изменений, внесённых в процессе выполнения запроса параллельными транзакциями. По сути запрос SELECT видит снимок базы данных в момент начала выполнения запроса. Однако SELECT видит результаты изменений, внесённых ранее в этой же транзакции, даже если они ещё не зафиксированы. Также два последовательных оператора SELECT могут видеть разные данные даже в рамках одной транзакции, если какие-то другие транзакции зафиксируют изменения после выполнения первого SELECT.

Частичная изоляция транзакций, обеспечиваемая в режиме Read Committed, приемлема для множества приложений. Этот режим быстр и прост в использовании, однако он подходит не для всех случаев. Приложениям, выполняющим сложные запросы и изменения, могут потребоваться более строго согласованное представление данных.

Для выбора нужного уровня изоляции транзакций используется команда SET TRANSACTION.

1. Чем Dockerfile отличается от Docker-образа? Когда нужно писать Dockerfile, а когда выгружать образ?

Dockerfile – это сценарий, который состоит из последовательности команд и аргументов, необходимых для создания образа. Такие сценарии упрощают развёртывание и процесс подготовки приложения к запуску.

Сначала Dockerfile определяет образ, на основе которого будет происходить сборка. Затем идёт ряд методов, команд и аргументов, которые создадут новый образ.

Содержимое Dockerfile передаётся демону Docker для сборки образа.

Образы Docker - это шаблоны для контейнеров. Они разработаны для того, чтобы быть эффективными и предлагать максимальное повторное использование, используя драйвер хранения файловой системы наложения.

Писать Dockerfile нужно в том случае, если нет возможности подключится к интернету (например сервер изолирован от общей сети), чтобы развернуть контейнер. А выгружать можно в том случае, когда необходимо сделать общедоступным контейнер для всех пользователей, которые могут скачать его через Docker Hub (при условии, что они в репозитории).

1. В чем отличие Dockerfile от docker-compose.yml?

Dockerfile – это подробный пошаговый сценарий для автоматизации сборки.

Docker-compose.yml — это как Dockerfile, но для распределенного приложения целиком.

При изучении основ Docker, в лабораторной работе столкнулись с созданием простейших приложений, работающих автономно, не зависящих, например, от внешних источников данных или от неких сервисов. На практике же подобные приложения — редкость. Реальные проекты обычно включают в себя целый набор совместно работающих приложений.

Например, когда создают веб-сайт, которому, для выполнения аутентификации пользователей, нужно подключиться к базе данных. Подобный проект может состоять из двух сервисов — того, что обеспечивает работу сайта, и того, который отвечает за поддержку базы данных.  
 Технология Docker Compose, если описывать её упрощённо, позволяет, с помощью одной команды, запускать множество сервисов.

И отличие заключает в том, что Dockerfile служит для создания простых приложений работающих автономно, не зависящих, например, от внешних источников данных или от неких сервисов. А docker-compose.yml –это совокупность различных контенеров виде сервисов без которых сложное приложение работать не будет.