**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

Галузинский И.А.

Проверил: к.т.н., доцент каф. МКиИТ

Махров С.С.

**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.

)

**Отчет по лабораторной работе №1**

**1.Придумать предметную область.**

Есть научный какой-то научный комплекс, который располагается

на большой территории, в нём работают большое количество

сотрудников. Сотрудники работают в разных отделах(подразделения)

(например, отдел систем безопасности, Web-технологий, бухгалтерии,

маркетинга и т.п.). У каждого работника есть должность, которую он занимает и зарплата этого сотрудника по умолчанию(но сотруднику могут назначить з\п отличную от установленной по умолчанию). Так же имеются разные офисы (например, простое офисное помещение с каким-то номером). Сотрудник может не иметь офиса, так как работает удалённо, но при этот он является сотрудником отдела. Некоторым сотрудникам предоставляют служебное авто(модель, фирма и ценна автомобиля закрепляется за id сотрудника). Так как это научный комплекс, то в нём есть лаборатории, в которых производят какие научные исследования. Каждое проведённое исследование записывают (какой отдел его проводил, затраты на проведения исследования, время потраченное на исследование(часах) и дату проведения).

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;

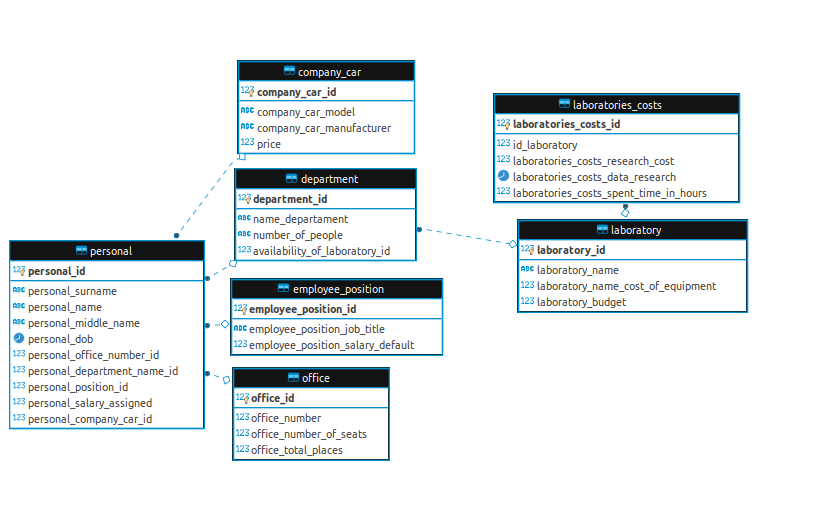


Рис. 1 — ER диаграмма

таблица personal(приложение 1 п.7) хранит в себе основную информацию о

персонале и ссылается на таблицы: personal(1 приложение п.1) в которой хранится информация об офисе, departament(1 приложение п.4 ) в которой находится информация отделе в котором работает персанал у отдела может быть и лаборатория,company\_car(1 приложение п.6 ), которой хранится название марка, модель и цена автомобиля, employee\_position(1 приложение п.5 ) в ней находится информация о зарплате по умолчанию у определённой должности и перечень должностей.

Таблица departament ссылается на неё(1 приложение п. 2) в ней хранятся данные о стоимости оборудования, названии лаборатории, бюджете. На таблицу laboratory ссылается таблица laboratories\_coast(приложение 1 п.3), так как у одной лаборатории может быть много исследований.

**3. Для наполнения использовались следующие тестовые данные:**

db.office(Приложение 2 п.1) — 3 объекта (номер офиса,колличество мест и количество всего мест в офисе)

laboratory(Приложение 2 п.2) – 3 объекта (название лаборатории, стоимость оборудования и бюджет лаборатории)

laboratories\_costs(Приложение 2 п.3) –4 объекта(id проводивший исследования лаборатории,затраченные средства на проведения исследования,дата проведения исследования,время затраченное на иследование)

department(Приложение 2 п.4) – 3 объекта (название отдела,количество людей в отделе и ссылка на id лаборатории в отделе, если у них такая имеется)

employee\_position(Приложение 2 п.5) –2 объекта (название должности,з\п по умолчанию)

company\_car(Приложение 2 п.6) – 3 объекта(модель машины,изготовитель машины, а так же цена автомобиля

personal(Приложение 2 п.7) – 9 обектов 4 из них являются исходящими(фамилия сотрудника, имя сотрудника,отчество сотрудника,дата роджения сотрудника,id офиса сотрудника,id отдела сотрудника,должность сотрудника,з\п которуб назначали сотруднику,id машины)

1. **Описание создания Dockerfile.**

В контейнере postgres есть каталог docker-entrypoint-initdb.d, на этапе сборки в него помещается скрипт sql (db.sql). Это нужно для того чтобы скрипт был выполнен на стадии создания контейнера. Скрипт включает в себя команды: создания базы данных, схемы, таблиц, а так же их наполнение данными таблиц созданных ранее. В скрипт db.sql необходимо прописать подлючения к базе данных командой: \connect db, после подключения в скрипте прописывается создание схемы и таблицы, а так же наполнение тестовыми данными таблицы.

Dockerfile (Приложение 3) состоит из 2 строчек

1- с каким образом (изображение) будет осуществлена работа

2- название скрита и директорию внутри образа/docker-entrypoint-initdb.d, куда будет помещён скрипт sql, который будут выполнены на старте.

который после запуска скрипт sql автоматически выполнится. Таким образом в результате запуска Dockerfile строится и наполняется база данных db данными.

1. **Описание выгрузки Docker-образа.**

Docker – образ извлечен и сохранен локально с помощью следующей команды:

sudo docker save -o lr1.tar lr1 . Но перед этим некоторым пользователя понадобится получить права супер пользователя, для того чтобы можно было использовать созданный архив.

Для загрузки образа и состояния контейнера из архива необходимо выполнить следующую команду:

docker load < lr1.tar

**6. CRUD запросы и результаты выполнения.**

Найдём всёх людей у которых в бд зарплата больше 60 000р

**select** \* **from** db.personal

**where** personal\_salary\_assigned >60000;

|  |  |
| --- | --- |
| Name | Value |
| personal\_id | 1 |
| personal\_surname | Иванов |
| personal\_name | Иван |
| personal\_middle\_name | Иванович |
| personal\_dob | 1977-01-12 |
| personal\_office\_number\_id | - |
| personal\_department\_name\_id | 2 |
| personal\_position | 2 |
| personal\_salary\_assigned | 1000000 |
| personal\_company\_car\_id | 2 |

Изменим заработную плату, у кого она больше 60 000 на 80 000:

Найдём людей и выведем их данные, которые имеют служебное авто

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob,

db.company\_car.company\_car\_model ,db.company\_car.company\_car\_manufacturer ,db.company\_car.price ,db.company\_car.company\_car\_id

**from** db.personal **join** db.company\_car

**on** personal.personal\_company\_car\_id = company\_car.company\_car\_id;

|  |  |  |
| --- | --- | --- |
| Name | Value1 | Value2 |
| personal\_id | 2 | 1 |
| personal\_name | Пётр | Иван |
| personal\_surname | Петоров | Иванов |
| personal\_middle\_name | Петрович | Иванович |
| personal\_dob | 1970-03-08 | 1977-01-12 |
| company\_car\_model | Niva 4x4 21214 | x5 |
| company\_car\_manufacturer | LADA | BNW |
| price | 600000 | 2500000 |
| company\_car\_id | 1 | 2 |

Усложним и добавим ещё отдел в котором работают люди, которые имеют служебное авто:

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob,

db.company\_car.company\_car\_model ,db.company\_car.company\_car\_manufacturer ,db.company\_car.price ,db.company\_car.company\_car\_id

,db.department.name\_departament ,db.department.number\_of\_people

**from** db.personal **join** db.company\_car

**on** personal.personal\_company\_car\_id = company\_car.company\_car\_id

**join** db.department **on** db.personal.personal\_department\_name\_id = department.department\_id

|  |  |  |
| --- | --- | --- |
| Name | Value1 | Value2 |
| personal\_id | 2 | 1 |
| personal\_name | Пётр | Иван |
| personal\_surname | Петоров | Иванов |
| personal\_middle\_name | Петрович | Иванович |
| personal\_dob | 1970-03-08 | 1977-01-12 |
| company\_car\_model | Niva 4x4 21214 | x5 |
| company\_car\_manufacturer | LADA | BNW |
| price | 600000 | 2500000 |
| company\_car\_id | 1 | 2 |
| name\_departament | машиностоительный | машиностоительный |
| number\_of\_people | 15 | 15 |

Найдём людей, которые работают в офисе

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob

,db.office.office\_number ,db.office.office\_number\_of\_seats ,db.office.office\_total\_places

**from** db.personal **join** db.office

**on** db.personal.personal\_office\_number\_id =db.office .office\_id ;

|  |  |
| --- | --- |
| Name | Value1 |
| personal\_id | 2 |
| personal\_name | Пётр |
| personal\_surname | Петоров |
| personal\_middle\_name | Петрович |
| personal\_dob | 1970-03-08 |
| office\_number | 44 |
| office\_number\_of\_seats | 210 |
| office\_total\_places | 1055 |

Найдём отделы, которые имеют лаборатории, выведем краткую информацию о них:

|  |  |  |
| --- | --- | --- |
| Name | Value | Value |
| department\_id | 1 | 2 |
| name\_departament | исследовательский в области медицины | машиностоительный |
| number\_of\_people | 10 | 15 |
| laboratory\_name | web | строительная |
| laboratory\_budget | 10000000 | 54321 |
| laboratory\_name\_cost\_of\_equipment | 100000 | 12345 |

**-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;

--выведем всесь персонал+ подробную информацию о них, кто имеет служебное авто марки lada

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob,

db.company\_car.company\_car\_model ,db.company\_car.company\_car\_manufacturer ,db.company\_car.price ,db.company\_car.company\_car\_id

,db.department.name\_departament ,db.department.number\_of\_people

**from** db.personal **join** db.company\_car

**on** personal.personal\_company\_car\_id = company\_car.company\_car\_id

**join** db.department **on** db.personal.personal\_department\_name\_id = department.department\_id

**where** db.company\_car.company\_car\_manufacturer ='LADA';

personal\_id|personal\_name|personal\_surname|personal\_middle\_name|personal\_dob|company\_car\_model|company\_car\_manufacturer|price |company\_car\_id|name\_departament |number\_of\_people|

|  |  |
| --- | --- |
| Name | Value |
| personal\_id | 2 |
| personal\_name | Пётр |
| personal\_surname | Петоров |
| personal\_middle\_name | Петрович |
| personal\_dob | 1970-03-08 |
| company\_car\_model | Niva 4x4 21214 |
| company\_car\_manufacturer | LADA |
| price | 600000 |
| company\_car\_id | 1 |
| name\_departament | Машиностоительный |
| number\_of\_people | 15 |

|

---выведем информацию о людях, которые не работают в офисе

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob,

db.company\_car.company\_car\_model ,db.company\_car.company\_car\_manufacturer ,db.company\_car.price ,db.company\_car.company\_car\_id

,db.department.name\_departament ,db.department.number\_of\_people

**from** db.personal **join** db.company\_car

**on** personal.personal\_company\_car\_id = company\_car.company\_car\_id

**join** db.department **on** db.personal.personal\_department\_name\_id = department.department\_id

**where** db.personal.personal\_office\_number\_id **is** **null** ;

|  |  |
| --- | --- |
| Name | Value |
| personal\_id | 1 |
| personal\_name | Иван |
| personal\_surname | Иванов |
| personal\_middle\_name | Иванович |
| personal\_dob | 1977-01-12 |
| company\_car\_model | x5 |
| company\_car\_manufacturer | BNW |
| price | 2500000 |
| company\_car\_id | 2 |
| name\_departament | Машиностоительный |
| number\_of\_people | 15 |

---выведем информацию о людях, которые работают в машиностоительнм депертаменте

**select** db.personal.personal\_id, db.personal.personal\_name,db.personal.personal\_surname ,db.personal.personal\_middle\_name ,db.personal.personal\_dob,

db.company\_car.company\_car\_model ,db.company\_car.company\_car\_manufacturer ,db.company\_car.price ,db.company\_car.company\_car\_id

,db.department.name\_departament ,db.department.number\_of\_people

**from** db.personal **join** db.company\_car

**on** personal.personal\_company\_car\_id = company\_car.company\_car\_id

**join** db.department **on** db.personal.personal\_department\_name\_id = department.department\_id

**where db.department.name\_departament='машиностоительный';**

|  |  |  |
| --- | --- | --- |
| Name | Value | Value |
| personal\_id | 1 | 2 |
| personal\_name | Иван | Пётр |
| personal\_surname | Иванов | Петоров |
| personal\_middle\_name | Иванович | Петрович |
| personal\_dob | 1977-01-12 | 1970-03-08 |
| company\_car\_model | x5 | Niva 4x4 21214 |
| company\_car\_manufacturer | BNW | LADA |
| price | 2500000 | 600000 |
| company\_car\_id | 2 | 1 |
| name\_departament | Машиностоительный | Машиностоительный |
| number\_of\_people | 15 | 15 |

--1 простой запрос на UPDATE к любой таблице по критериям, указанным в блоке WHERE;

--уменьшим з\п всем у кого она выше 60 000

**update** db.personal

**set** personal\_salary\_assigned = 80000

**where** personal\_salary\_assigned >= 60000;

**select** \* **from** db.personal ;

|  |  |
| --- | --- |
| Name | Value |
| personal\_id | 1 |
| personal\_surname | Иванов |
| personal\_name | Иван |
| personal\_middle\_name | Иванович |
| personal\_dob | 1977-01-12 |
| personal\_office\_number\_id | **null** |
| personal\_department\_name\_id | 2 |
| personal\_position | 2 |
| personal\_salary\_assigned | Было:1 000 000 стало: 80 000 |
| personal\_company\_car\_id | 2 |

--1 сложный запрос на UPDATE -обновление таблицы A в зависимости от данных в таблице B (придумать критерии);

-- повысить з\п кому выданна машина от компании производства фирмы LADA

**UPDATE** db.personal

**set** personal\_salary\_assigned = personal\_salary\_assigned+123456

**from** db.company\_car

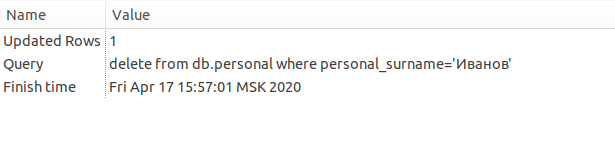
**where** db.personal.personal\_company\_car\_id =db.company\_car.company\_car\_id

**and** db.company\_car.company\_car\_manufacturer ='LADA';

|  |  |  |
| --- | --- | --- |
| Name | Было | стало |
| personal\_id | 2 | 2 |
| personal\_surname | Петоров | Петоров |
| personal\_name | Пётр | Пётр |
| personal\_middle\_name | Петрович | Петрович |
| personal\_salary\_assigned | 10000 | 133456 |

-- 1 простой запросы на DELETE - к любой таблице по критериям, указанным в блоке WHERE;

**delete** **from** db.personal **where** personal\_surname='Иванов' ;



-- • 1 сложный запрос на DELETE - удаление записей из таблицы A в зависимости от данных в таблице B (придумать критерии);

--удалим сотрудников, которые работают в офисе 2

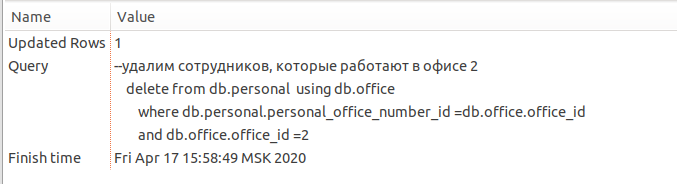
**delete** **from** db.personal **using** db.office

**where** db.personal.personal\_office\_number\_id =db.office.office\_id

**and** db.office.office\_id =2;

**select** \* **from** db.personal ;

был удалён Петров

****

**7. select for update.**

Найдём людей, которые не имеет офиса, временно заблокируем его изменение в бд.

**select** db.personal.personal\_id , db.personal.personal\_name,db.personal.personal\_surname,db.personal.personal\_middle\_name

**from** db.personal

**where** db.personal.personal\_office\_number\_id **is** **null**

--limit 2

**for update**

# 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 и освоен процесс развертывания Postgres в контейнере Docker.

Приложение 1

(В приложении указать листинг DDL. На приложение должна быть ссылка в отчете)

1. Таблица office

--офис в котором работают сотрудники

**CREATE** **TABLE** db.office (

office\_id **serial** **primary** **key**--id

,office\_number **integer** **NOT** **null**-- номер офиса в котором работает сотрудник

,office\_number\_of\_seats **integer** **NOT** **null**-- количество колличество мест

,office\_total\_places **int** **NOT** **null**--всего мест в офисе

);

2. Таблица laboratory

-- лаборатория, если есть

**CREATE** **TABLE** db.laboratory (

laboratory\_id **serial** **primary** **key**--id

,laboratory\_name **varchar**(50) **not** **Null**--название лаборатории

,laboratory\_name\_cost\_of\_equipment **int**--стоимость оборудования

,laboratory\_budget **int** **NOT** **null** --бюджет лаборатории

);

3. Таблица laboratories\_costs

-- затраты лаборатории

**CREATE** **TABLE** db.laboratories\_costs (

laboratories\_costs\_id **serial** **primary** **key**--id

,id\_laboratory **int** **references** db.laboratory(laboratory\_id)--id проводивший исследования лаборатории

,laboratories\_costs\_research\_cost **int** --затраченные средства на проведения исследования

,laboratories\_costs\_data\_research **date** **NOT** **null**--дата проведения исследования

,laboratories\_costs\_spent\_time\_in\_hours **int4** **NOT** **null**--время затраченное на иследование

);

4. Таблица department

--отдел в котором работают сотрудники

**CREATE** **TABLE** db.department (

department\_id **serial** **primary** **key**--id

,name\_departament **varchar**(50) **NOT** **null**--название отдела

,number\_of\_people **varchar**(50) --количество людей в отделе

,availability\_of\_laboratory\_id **int** **references** db.laboratory(laboratory\_id) --id лаборатории в отделе

);

5. Таблица employee\_position

-- должность сотрудника

**CREATE** **TABLE** db.employee\_position (

employee\_position\_id **serial** **primary** **key**--id

,employee\_position\_job\_title **varchar**(50) **NOT** **null**--название должности

,employee\_position\_salary\_default **int** **NOT** **null**-- з\п по умолчанию

);

6. Таблица company\_car

-- служебное авто

**CREATE** **TABLE** db.company\_car (

company\_car\_id **serial** **primary** **key**--id

,company\_car\_model **varchar**(50) **null** --модель машины

,company\_car\_manufacturer **varchar** **not** **null** --изготовитель машины

,price **int** **NOT** **null**--цена автомобиля

);

7. Таблица personal

**CREATE** **TABLE** db.personal (

personal\_id serial **primary** **key**--id

,personal\_surname **varchar**(50) **NOT** **null**--фамилия сотрудника

,personal\_name **varchar**(30) **NOT** **null**-- имя сотрудника

,personal\_middle\_name **varchar**(40) --отчество сотрудника

,personal\_dob **date** **NOT** **null**--дата роджения сотрудника

,personal\_office\_number\_id **integer** **references** db.office(office\_id ) -- id офиса сотрудника

,personal\_department\_name\_id **int** **references** db.department(department\_id ) **NOT** **null**--id отдела сотрудника

,personal\_position\_id **int** **references** db.employee\_position(employee\_position\_id) **NOT** **null**--должность сотрудника

,personal\_salary\_assigned **integer** **NOT** **null**-- з\п которуб назначали сотруднику

,personal\_company\_car\_id **integer** **references** db.company\_car(company\_car\_id ) -- id машины

Приложение 2

(В приложении указать листинг DML. На приложение должна быть ссылка в отчете)

**1. insert** **into** db.office(

office\_number --номер офиса в котором работает сотрудник

,office\_number\_of\_seats --количество колличество мест

,office\_total\_places --всего мест в офисе

)**values**

(1,20,55),

(44,210,1055);

**2. insert** **into** db.laboratory(

laboratory\_name --название лаборатории

,laboratory\_name\_cost\_of\_equipment --стоимость оборудования

,laboratory\_budget --бюджет лаборатории

)**values**

('web',100000,10000000),('строительная',12345,54321);

**3. insert** **into** db.laboratories\_costs (

id\_laboratory --id проводивший исследования лаборатории

,laboratories\_costs\_research\_cost --затраченные средства на проведения исследования

,laboratories\_costs\_data\_research --дата проведения исследования

,laboratories\_costs\_spent\_time\_in\_hours --время затраченное на иследование

)**values**

(2,90, '2013-06-01',100),(1,10, '2019-07-11',133);

**4. insert** **into** db.department (

name\_departament --название отдела

,number\_of\_people --количество людей в отделе

,availability\_of\_laboratory\_id --id лаборатории в отделе

)**values**('исследовательский в области медицины',10,2),

('машиностоительный',15,**Null**);

**5. insert** **into** db.employee\_position (

employee\_position\_job\_title --название должности

,employee\_position\_salary\_default -- з\п по умолчанию

)**values**('младший научный сотрудник',1000),('директор',5789);

**6. insert** **into** db.company\_car (

company\_car\_model --модель машины

,company\_car\_manufacturer --изготовитель машины

,price --цена автомобиля

)**values**('Niva 4x4 21214','LADA' ,600000),('x5','BNW' ,2500000),('Q6','AUDI' ,1000);

**7. insert** **into** db.personal (

personal\_surname --фамилия сотрудника

,personal\_name --имя сотрудника

,personal\_middle\_name --отчество сотрудника

,personal\_dob --дата роджения сотрудника

,personal\_office\_number\_id --id офиса сотрудника

,personal\_department\_name\_id --id отдела сотрудника

,personal\_position\_id --id должность сотрудника

,personal\_salary\_assigned --з\п которуб назначали сотруднику

,personal\_company\_car\_id --id машины

)**values**('Иванов','Иван','Иванович','1977-01-12',**null**,2,2,1000000,2),

('Петоров','Пётр','Петрович','1970-03-08',2,2,1,10000,1);

Приложение 3

(В приложении указать листинг полученного Dockerfile)

FROM postgres

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

# 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?

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 [базами данных](https://ru.wikipedia.org/wiki/База_данных): создание ([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.