ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Кафедра вычислительной техники

КУРСОВАЯ РАБОТА «Использование NoSQL (not only SQL) баз данных»

ПО ДИСЦИПЛИНЕ «СИСТЕМЫ БАЗ ДАННЫХ»

Выполнили: Плюхин Дмитрий Алексеевич

Останина Анастасия Александровна

Группа: P3317

Преподаватель: Беликов Павел Андреевич

\

Санкт-Петербург

2018

Постановка задачи

Предметная область – Очень Большой Космический корабль.

Информация, хранимая в системе – данные, предназначенные для описания корабля, отправляющегося на некоторую планету не-Марс.

Тип базы данных	Содержимое базы данных
Документо-ориентированная	Описание корабля, профили команды
Колоночная	История операций, бортовой журнал
Графовая	Описание взаимоотношений между участниками экспедиции

Описание разработанной системы

TABLE OF CONTENTS

Home.md

The Document

The document describes the main points needed for base understanding of the project. It is outlined, what is necessary for successful running the system as well as how the project is organised internally.

The system

Information system, containing data about activity of hypotetical spaceship, which is going to the planet not-Mars.

Repository on github (https://github.com/ostaninanastya/space_ship)

Project structure

`/api` contains mappers and manipulators for making it possible to interact with system via GraphQL (http://graphql.org/learn/) as well as web-server, written on node.js (https://nodejs.org/en/) in order to simplify testing

`/clustering` (versions, which support clustering are located on the branches **clustering** and **ilm**) contains config files used for setting up mongodb (https://www.mongodb.com/), cassandra (http://cassandra.apache.org/) and neo4j (https://neo4j.com/) clusters

`/connectors` contains sets of functions for setting up connection to the databases using vendor's libraries

`/generation` contatins data generators written in java for the databases; all of the generators use dummymaker (https://github.com/GoodforGod/dummymaker) library

`/logbook` contains code which is relevant for collections from the 'logbook' section, which represent journals

`/recital` contains code which is relevant for collections from the 'recital' section, which represent basic info about the ship and it's command

`/relations` contains code which is relevant for collections from the 'relations' section, which describes relations between people on the ship and introduces few extra entities

`/store` (on the branch **ilm**) contains code which is relevant for the base data container in the ilm architecture (see wiki (https://github.com/ostaninanastya/space_ship/wiki) to find out more about that) - for these purposes MySQL (https://www.mysql.com/) was being used

`/transporters` (on the branch **ilm**) contains modules, dedicated for exchanging data between databases in the ilm architecture

1.-Prerequisites.md

Прежде всего, для нормальной работы всех скриптов необходимо задать в переменной окружения **SPACE_SHIP_HOME** путь к папке приложения.

mongo server

Для установки на Ubuntu:

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.6 multiverse" | sudo tee /etc/ap
sudo apt-get update
sudo apt-get install -y mongodb-org
```

cassandra server

Для установки из debian packages (**изменить версию с 311x на актуальную**)

```
echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources
curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
sudo apt-get update
sudo apt-key adv --keyserver pool.sks-keyservers.net --recv-key A278B781FE4B2BDA
sudo apt-get install cassandra
```

neo4j server

Установка производится так:

```
wget --no-check-certificate -O - https://debian.neo4j.org/neotechnology.gpg.key | sudo apt-key add -
echo 'deb http://debian.neo4j.org/repo stable/' | sudo tee /etc/apt/sources.list.d/neo4j.list
sudo apt update
sudo apt install neo4j
```

cassandra-driver

Модуль для python, с помощью которого идет взаимодействие с БД

Устанавливается командой (может занять несколько минут)

```
pip install cassandra-driver
```

pymongo

Модуль для python, с помощью которого идет взаимодействие с базой данных, содержащей информацию о корабле (работающей на технологии Mongo DB) для обеспечения консистентности хранимых данных при их изменении

Устанавливается командой

```
pip install pymongo
```

py2neo

Модуль для python, с помощью которого идет взаимодействие с базой данных, содержащей информацию об организации деятельности экипажа (работающей на технологии neo4j) для обеспечения консистентности хранимых данных при их изменении

Устанавливается командой

```
pip install py2neo
```

neomodel

ODM для python, с помощью которого идет взаимодействие с neo4j

Устанавливается командой

```
pip install neomodel
```

maven

Система сборки для создания jar-архива с пользовательскими функциями для neo4j

Устанавливается например так

```
sudo apt-get install maven
```

node js

Платформа для запуска веб-сервера, с помощью которого можно можно пользоваться api из браузера

graphene

Библиотека, упрощающая реализацию api на GraphQL

```
pip3 install graphene
```

2.-Entities.md

Recital

Информация о составе корабля и его экипажа

Поскольку каждому экземпляру сущности автоматически присваивается object id, и именно он используется в дальнейшем для ссылки на экземпляры сущностей в том числе и из других баз данных, далее в описании сущностей он указываться не будет.

- department - сущность для представления департамента в информационной системе

property	**data type**	**description**
name	string	Название департамента
hrefToVkCommunity	string	Ссылка на сообщество вконтакте

- people - сущность для представления члена экипажа корабля

property	**data type**	**description**
name	string	Имя члена экипажа
surname	string	Фамилия члена экипажа
patronymic	string	Отчество члена экипажа
department	object id	Ссылка на департамент в котором числится член экипажа
phoneNumber	string	Номер телефона
specialization	object id	Ссылка на специализацию, присвоенную члену экипажа

- properties - материальное имущество корабля (**технические средства**, **мебель**, **средства вооружения** и т.д.)

property	**data type**	**description**
name	string	Название предмета
type	object id	Ссылка на тип, к которому принадлежит элемент имущества
dateOfAdmission	date	Дата поступления на корабль, внесения в списки имущества и, соответственно, введения в эксплуатацию
comissioningDate	date	Дата прекращения эксплуатации элемента имущества, исключения из списков корабля, и, в некоторых случаях, выброса во внешнюю среду
department	object id	Ссылка на департамент в котором числится предмет

- propertyTypes - типы материального имущества корабля

property	**data type**	**description**
name	string	Название типа имущества (**laptop**, **table**, **laser gun**, **big fragging gun**, etc)
description	string	Описание типа имущества

- specializations - специальности членов экипажа

property	**data type**	**description**
name	string	Название специальности (**electrician**, **doctor**, **first engineer**, **second engineer**, etc)

- states - возможные состояния систем корабля

property	data type	description
name	string	Название состояния (**ready**, **repair**, **working**, etc)
description	string	Описание состояния

- systems - системы, имеющиеся на корабле

property	data type	description
name	string	Название системы (**fuel system**, **main engine**, **spare engine**, **main thrusters**, **spare thrusters**, **power system**, etc)
type	object id	Тип системы корабля
serialNumber	double	Серийный номер, присвоенный заводом-изготовителем
dateOfLaunch	date	Дата запуска системы
dateOfLastChecking	date	Дата последней проверки системы
personInCharge	object id	Ссылка на сущность, представляюшую человека, ответственного за обеспечение нормального режима работы системы
state	object id	Ссылка на состояние, присвоенное системе

- systemTypes - типы систем, имеющихся на корабле

property	data type	description
name	string	Название типа систем (**fuel**, **engine**, **thrusters**, **hydraulic**, etc)
description	string	Описание типа системы корабля

- sensors - источники данных об окружающей обстановке

property	data type	description
name	string	Название сенсора (**MINAS_MORGUL_T400**, **STADDLE_N23**, **VALMAR_17**, etc)
location	object id	Ссылка на объект, представляюший расположение сенсора

- boats - информационные модели судов меньшего размера, хранящихся на корабле и предназначенных для следования к местам совершения операций во внешней среде без изменения курса корабля

property	data type	description
name	string	Название судна
capacity	integer	Количество человек, которое может принять судно

- locations - возможные расположения сенсоров корабля

property	data type	description
name	string	Название места расположения, которое должно давать представление о том, где находится сенсор (**top edge**, **bottom edge**, **left side**, etc)

Logbook

Журналы с данными о работе корабля

- system_test - предназначена для записи результатов тестирования систем корабля

property	data type	description
date	date	Дата, для которой актуально приведенное состояние системы
time	time	Временная отметка, для которой актуально приведенное состояние системы

property	data type	description
system id	12 bytes	Идентификатор системы, состояние которой было зафиксировано
result	int	Результат теста, может принимать значения от 0 до 100

- control_action - описывает команду, когда-либо выданную системе управления кораблем (например, повернуть налево или снять показания датчиков космического излучения по левому борту)

property	data type	description
date	date	Дата, определяющая момент получения команды
time	time	Временная отметка, определяющая момент получения команды
mac_address	6 bytes	MAC-адрес устройства, выдавшего команду
user_id	12 bytes	Идентификатор пользователя, выдавшего команду
command	text	Выданная команда (например, **get**)
params	text	Параметры, уточняющие, к какому результату должна привести команда (например, **--sensor=MINAS_MORGUL T400 --value_name=cold_dark_matter_concentration**)
result	text	Результат работы команды (например, **23.235715395881783 TeV**)

- position - содержит историю перемещения корабля в космическом пространстве

property	data type	description
date	date	Дата, определяющая момент получения команды
time	timestamp	Временная отметка, определяющая момент получения команды
x	double	Координата корабля по оси OX, которая направлена от Земли в сторону планеты не-Марс
y	double	Координата корабля по оси OY, которая перпендикулярна оси OX и направлена в сторону созвездия Кассиопея
z	double	Координата корабля по оси OZ
speed	double	Линейная скорость корабля
attack_angle	double	Угол между продольной осью корабля и осью OX
direction_angle	double	Угол между проекцией продольной осью корабля на плоскость XOY и осью OX

- sensors_data - содержит данные о состоянии внешней среды, поставляемые сенсорами, расположенными по периметру корабля

property	data type	description
date	date	Дата, определяющая момент получения данных от сенсора
time	time	Временная отметка, определяющая момент получения данных от сенсора
source_id	12 bytes	Идентификатор сенсора, предоставившего данные
event	text	Наименование события, ставшего причиной фиксирования данных (например, **request**)
value_name	text	Название измеренной величины (например, **space_radiation**)
value	double	Значение измеренной величины
units	text	Единицы измерения (например, **eV** - сокрашение от электрон-вольт

- shift_state - хранит обстановку в области, доверенной некоторой смене, на данный момент времени

property	data type	description
date	date	Дата, для которой характерны данные о смене
time	time	Временная отметка, для которой характерны данные о смене

property	data type	description
shift_id	16 bytes	Идентификатор смены, для которой характерно зафикисированное состояние на некоторый момент времени
warning_level	text	Уровень опасности в районе, доверенном смене (может принимать значения **lowest**, **low**, **medium**, **heigh**, **highest**)
remaining_cartridges	double	Оставшиеся патроны (в процентах)
remaining_air	double	Оставшийся воздух (в процентах)
remaining_electricity	double	Оставшееся электричество (в процентах)
comment	text	Необязательный комментарий от работников смены

- operation_state - содержит описание состояния операции на некоторый момент времени

property	data type	description
date	date	Дата, для которой характерны данные об операции
time	time	Временная отметка, для которой характерны данные об операции
boat_id	12 bytes	Необязательный идентификатор машины, зарезервированной для выполнения операции
operation_id	16 bytes	Идентификатор операции
operation_status	text	Состояние операции на данный момент времени (например, **detaching_from_ship**)
distance_to_the_ship	double	Расстояние от судна, предназначенного для операции, до корабля
zenith	double	Зенитный угол, задающий положение судна, предназначенного для операции, относительно корабля
azimuth	double	Азимутальный угол, задающий положение судна, предназначенного для операции, относительно корабля
hydrogenium	double	Процентное содержание водорода в атмосфере / космической пыли вокруг команды, выполняющей операцию
helium	double	Процентное содержание гелия в атмосфере / космической пыли вокруг команды, выполняющей операцию
...	double	115 полей, содержащих данные о процентном содержании химических элементов в атмосфере / космической пыли вокруг команды, выполняющей операцию
oganesson	double	Процентное содержание оганесона в атмосфере / космической пыли вокруг команды, выполняющей операцию
comment	double	Необязательный комментарий от участников команды, выполняющей операцию

Relations

Информация о служебных взаимоотношениях экипажа

- person - представление члена экипажа в рамках графической модели данных

property	data type	description
ident	two ints [1]	Идентификатор члена экипажа, соответствующий 12 - байтному идентификатору в базе данных, содержащей информацию о корабле (основанной на Mongo DB)
controlled	reference	Связь с сущностью Department, отображающая отношение главенствования над департаментом (DIRECTOR)
executor	reference	Связь с сущностью Operation, отображающая отношение исполнения операции (EXECUTOR)
headed	reference	Связь с сущностью Operation, отображающая отношение руководства операцией (HEAD)
worker	reference	Связь с сущностью Shift, отображающая отношение рядового участника смены (WORKER)
chief	reference	Связь с сущностью Shift, отображающая отношение ответственного за смену (CHIEF)

- department - представление департамента в рамках графической модели данных

property	data type	description
ident	two ints [1]	Идентификатор департамента
shifts	reference	Связь с сущностью Shift, отображающая отношение организации смены департаментом (INCORPORATION)
operations	reference	Связь с сущностью Operation, отображающая отношение организации операции департаментом (INCORPORATION)
controller	reference	Связь с сущностью Person, отображающая отношение руководства департаментом (DIRECTOR)

- operation - сущность, содержащая основные сведения о планируемой или уже осуществленной операции

property	data type	description
ident	16 bytes	Идентификатор операции
name	text	Название операции
start	datetime	Дата и время начала операции
end	datetime	Дата и время окончания операции
department	reference	Связь с сущностью Operation, отображающая отношение организации операции департаментом (INCORPORATION)
requirement	reference	Связь с сущностью Requirement, отображающая отношение использования набора требований по специализации личного состава для операции (USER)
persons	reference	Связь с сущностью Operation, отображающая отношение исполнения операции (EXECUTOR)
head	reference	Связь с сущностью Operation, отображающая отношение руководства операцией (HEAD)

- shift - сущность, содержащая основные сведения о прошедшей или предстоящей смене

property	data type	description
ident	16 bytes	Идентификатор смены
start	datetime	Дата и время начала смены
end	datetime	Дата и время окончания смены
department	reference	Связь с сущностью Operation, отображающая отношение организации операции департаментом (INCORPORATION)
requirement	reference	Связь с сущностью Requirement, отображающая отношение использования набора требований по специализации личного состава для смены (USER)
persons	reference	Связь с сущностью Operation, отображающая отношение исполнения операции (EXECUTOR)
head	reference	Связь с сущностью Operation, отображающая отношение руководства операцией (HEAD)

- requirement - набор требований к подготовке личного состава персонала для выполнения операции или образования смены

property	data type	description
ident	16 bytes	Идентификатор набора
name	text	Название набора
content	array of jsons	Массив объектов json, каждый из которых имеет по крайней мере два поля: **ident** - идентификатор специальности из mongo db [1] и **quantity** - количество сотрудников с такой специальностью
operations	reference	Связь с сущностью Operation, отображающая отношение использования операцией набора требований (USER)
shifts	reference	Связь с сущностью Shift, отображающая отношение использования операцией набора требований (USER)

Note about two ints

Значения в базе данных neo4j, соответствующие индексам mongodb хранятся в виде пар значений целочисленного типа. Для перевода такой пары обратно в 12-байтный индекс, необходимы вычислить значение

```
item[0] * 2^48 + item[1]
```

И интерпретировать его как 12-байтный индекс

Например, если имеем mongo - идентификатор

```
5abfdba6ee6b7f5eec83a1ca
```

То можем его отобразить в целочисленном представлении

```
280855929930666802948931342 82
```

Или, если разбить на два числа, то получится

```
(99780070403691, 140045671702986)
```

Обратное преобразование выглядит как

```
(int(math.pow(2, 48)) * 99780070403691 + 140045671702986).to_bytes(12, byteorder='big').hex()
```

Описанный механизм работает подобно механизму преобразования даты и времени из используемого ODM в число с плавающей точкой для сохранения в базе данных.

Преобразование может быть сделано при помощи пользовательской функции space_ship.get_hex_ident - для этого необходимо скопировать файл `neo4j_functions-1.0.0.jar` из папки relations/functions/target в папку plugins, соответствующую серверу neo4j (например, `/var/lib/neo4j/plugins`). Далее функцию можно вызывать в предложениях по выборке данных, например, так:

```
match (n:Person) return space_ship.get_hex_ident(n.ident);
```

Перекомпиляция исходного кода и запуск тестов делаются с помощью команды

```
mvn clean package
```

3.-Running-databases.md

Mongo DB

Запуск сервера

```
sudo service mongod start
```

Logbook

Запуск сервера

```
sudo service cassandra start
```

Выполнить команду, которая должна завершиться выводом информации о сервере с зеленым словом 'active'

```
sudo service cassandra status
```

Для проверки статуса кластера выполнить

```
sudo nodetool status
```

Должна вывести результат, начинающийся с сочетания **UN** (Up and Normal) в случае успеха

Relations

Выполнить команду

```
sudo service neo4j start
```

Веб-интерфейс станет доступен (стандартно) по адресу

```
http://localhost:7474
```

4.-Filling-databases.md

Mongo

Находясь в папке `/generation/dummyMarker` , выполнить

```
mvn install
mvn exec:java
```

Для перекомпиляции и последующей генерации json файлов. Далее, перейдя в папку `/recital` , выполнить

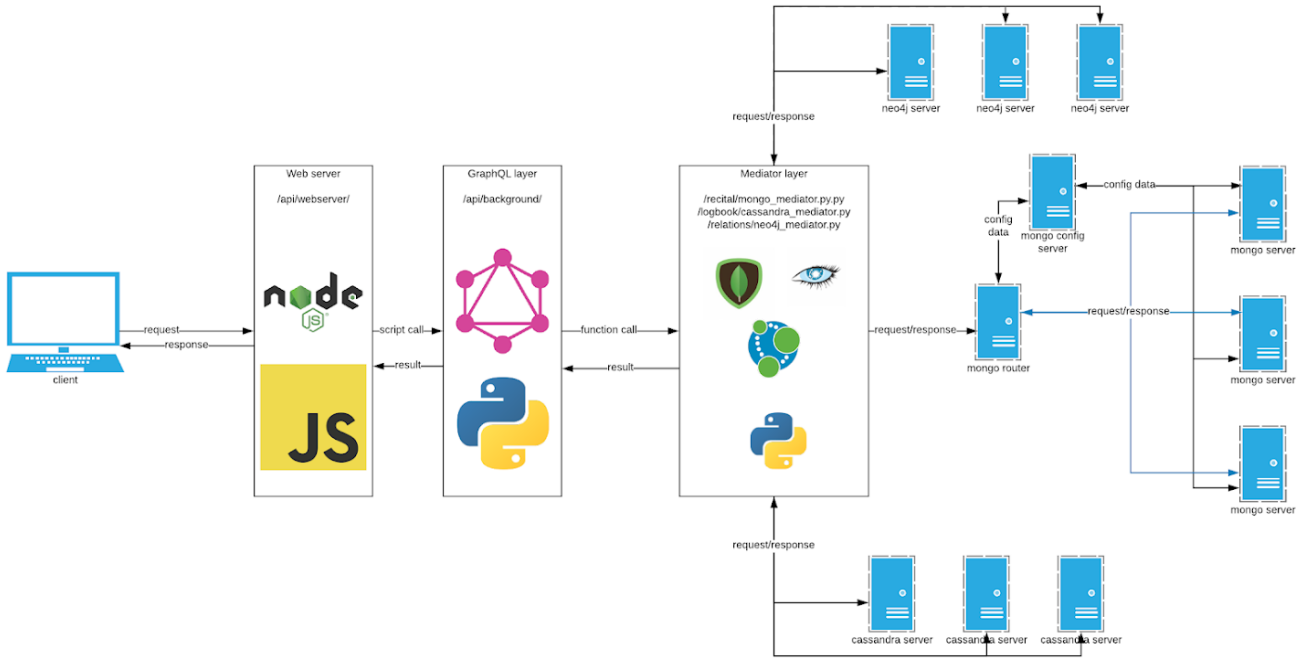
```
python fill.py
```

Для заполнения базы данных mongo

5.-API.md

Architecture overview

View in full size (https://lh3.googleusercontent.com/Fo9hQuOui-LCtnZf5IP4TDo4_kyVp0arwSRpgfDoEjeyc2ptAKtnxPp2is5xZ2TKSUZpgw4ZwZPko9NxAcr4qqNLiN9R8ohaRNk8k6P0ciYDwfXR4i2u_Nvd_4xGjy3Wyvh7VlHsu-r597CM_4N40J--doOx8fQPYZdXdHlwag7jqBibXPv2y8gYXKorc7Vpm0sdMqEXLj1P1PFGxIv40iygTr-V41jKBB81OlymfR7Yawsv74YBc1-2E5F-TgaT1PHFo0jW3Qp9qTr5OxphsJ7V3kA1-qPlCKJ9jR4tYEfBgshJo9MPJSfNm-_e0kE46a9hkHy6b0puFelbk9EKNBNaojkdpZhCobxLQU34BPtgjb9R7eevIipiUOgFjSNiVd7ce0OCsmvtnEbufAEQpWDUCu8gKPtS-WbZbKv7p2pbHi_uifNEf9ORo0ajzCM9Ezx_92mFoRPX7dJ-GCrHVwNIv5_ki8ikiVIyWs09H5cyhG1fyS6C-vL9A9fAvkyW5c045W7OJegH7bDHP3b1tNVKYNV4xavjkf-A9sKwuNcwPLRlY-IJaj8L8KknzHGK-wHriU4SVfwXZ0ix18oh6lld1NTWuHtQYXqm4eZEqtO0dUWYF7x3JJ0V52D-U2IFH-Lt-m_bV0SV_1VQAWLzekdOXeMVVswO=w1492-h764-no)



Web API

CRUD

Primarily it is considered that the application is being accessed via web interface. It consists of two main parts: `GET` method interface and `POST` method interface.

GET

In order to use the application via `GET` interface the queries of the view should be used:

`http://localhost:1881/api/create/person/fields=ok&where=name:'John',surname:'White'` for **creation**

`http://localhost:1881/api/people/fields=name,surname&where=name:'John'` for **reading**

`http://localhost:1881/api/people/fields=name,surname&where=name:'John'&set=specialization:5accfcc5ee18bbcfe8035add` for **updating**

`http://localhost:1881/api/delete/person/fields=ok&where=name:'John'` for **deleting**

POST

`POST` queries are significantly differs from the `GET` queries - in contrast to seconds they use **html query body** encoded in json to represent query parameters. All the queries, described below, have to be sent to the address `http://localhost:1881/api/` .

for **creation**:

```
{
    "operation": "create",
    "entity": "person",
    "fields" : ["ok"],
    "where": {
        "name": "John",
            "surname": "White"
    }
}
```

for **reading**:

```
{
    "operation": "read",
    "entity": "requirements",
    "fields" : ["name", {
        "content" : [{
            "specialization": ["name"]
        }, "quantity"]
    }]
}
```

for **updating**:

```
{
    "operation": "update",
    "entity": "requirements",
    "fields" : ["ok"],
    "where": {
        "name": "programmersa"
    },
    "set": {
        "content": "5aeddb09d678f433ec9b286c:25,5aeddb19d678f4341fc29b82:45"
    }
}
```

for **deleting**:

```
{
    "operation": "remove",
    "entity": "requirements",
    "fields" : ["ok"],
        "where": {
        "name": "programmersa"
    }
}
```

ILM

Moreover, there is a possibility of getting content from any layer of the ilm model (see ilm (https://github.com/ostaninanastya/space_ship/wiki/7.-Ilm) section to find out more about that). For example, to get all people entries from the hot-cache layer, just send a request with body

```
{
    "layer": "hot-cache",
    "entity": "people"
}
```

to the address `http://localhost:1881/data/` .

GraphQL

Взаимодействие с базой данных производится при помощи GraphQL, для этого можно вызывать скрипт main.py из папки /api/background, в этом случае вызов скрипта будет иметь вид

```
python3 test.py 'mutation Mutation{ createSensor(name:"rohan",location:"5aca531ed532b43d2c7d0f34"){ sensor{name} }}'
```

Далее приведен общий список объектов GraphQL, соответствующих описанным ранее сущностям базы данных. Этот список формируется сервером по адресу http://localhost:1881/docs/ (http://localhost:1881/docs/)

```
=== Queries ===

= Entities =

    position{
        date: String
        time: String
        x: Float
        y: Float
        z: Float
        speed: Float
        attackangle: Float
        directionangle: Float
    }

    controlaction{
        date: String
        time: String
        macaddress: String
        userid: String
        user: PersonMapper
        command: String
        params: String
        result: String
    }

    person{
        id: String
        name: String
        department: DepartmentMapper
        directing: List
        chiefed: List
        worked: List
        headed: List
        executed: List
        specialization: SpecializationMapper
        commands: List
        supervised: List
    }

    department{
        id: String
        name: String
        director: PersonMapper
        vk: String
        properties: List
        people: List
    }

    property{
        id: String
        name: String
        type: PropertyTypeMapper
        admission: String
        comissioning: String
        department: DepartmentMapper
    }

    propertytype{
        id: String
        name: String
        description: String
        properties: List
    }

    shift{
        id: String
        start: String
        end: String
        chief: PersonMapper
        workers: List
        requirements: List
    }

    requirement{
        id: String
        name: String
        content: List
        shifts: List
        operations: List
    }
```

```
requiremententry{
    specialization: SpecializationMapper
    quantity: Int
}

specialization{
    id: String
    name: String
    people: List
}

operation{
    id: String
    name: String
    start: String
    end: String
    head: PersonMapper
    executors: List
    requirements: List
}

system{
    id: String
    name: String
    serialnumber: Float
    launched: String
    checked: String
    state: SystemStateMapper
    supervisor: PersonMapper
    type: SystemTypeMapper
}

systemstate{
    id: String
    name: String
    description: String
    systems: List
}

systemtype{
    id: String
    name: String
    description: String
    systems: List
}

systemtest{
    date: String
    time: String
    system: SystemMapper
    systemid: String
    result: Int
}

operationstate{
    date: String
    time: String
    boatid: String
    boatname: String
    operationid: String
    operation: OperationMapper
    operationstatus: String
    distancetotheship: Float
    zenith: Float
    azimuth: Float
    hydrogenium: Float
    helium: Float
    lithium: Float
    beryllium: Float
    borum: Float
    carboneum: Float
    nitrogenium: Float
    oxygenium: Float
    fluorum: Float
    neon: Float
    natrium: Float
    magnesium: Float
    aluminium: Float
    silicium: Float
    phosphorus: Float
```

sulfur: **Float**
chlorum: **Float**
argon: **Float**
kalium: **Float**
calcium: **Float**
scandium: **Float**
titanium: **Float**
vanadium: **Float**
chromium: **Float**
manganum: **Float**
ferrum: **Float**
cobaltum: **Float**
niccolum: **Float**
cuprum: **Float**
zincum: **Float**
gallium: **Float**
germanium: **Float**
arsenicum: **Float**
selenium: **Float**
bromum: **Float**
crypton: **Float**
rubidium: **Float**
strontium: **Float**
yttrium: **Float**
zirconium: **Float**
niobium: **Float**
molybdaenum: **Float**
technetium: **Float**
ruthenium: **Float**
rhodium: **Float**
palladium: **Float**
argentum: **Float**
cadmium: **Float**
indium: **Float**
stannum: **Float**
stibium: **Float**
tellurium: **Float**
iodium: **Float**
xenon: **Float**
caesium: **Float**
barium: **Float**
lanthanum: **Float**
cerium: **Float**
praseodymium: **Float**
neodymium: **Float**
promethium: **Float**
samarium: **Float**
europium: **Float**
gadolinium: **Float**
terbium: **Float**
dysprosium: **Float**
holmium: **Float**
erbium: **Float**
thulium: **Float**
ytterbium: **Float**
lutetium: **Float**
hafnium: **Float**
tantalum: **Float**
wolframium: **Float**
rhenium: **Float**
osmium: **Float**
iridium: **Float**
platinum: **Float**
aurum: **Float**
hydrargyrum: **Float**
thallium: **Float**
plumbum: **Float**
bismuthum: **Float**
polonium: **Float**
astatum: **Float**
radon: **Float**
francium: **Float**
radium: **Float**
actinium: **Float**
thorium: **Float**
protactinium: **Float**
uranium: **Float**
neptunium: **Float**
plutonium: **Float**
americium: **Float**
curium: **Float**

```
        berkelium: Float
        californium: Float
        einsteinium: Float
        fermium: Float
        mendelevium: Float
        nobelium: Float
        lawrencium: Float
        rutherfordium: Float
        dubnium: Float
        seaborgium: Float
        bohrium: Float
        hassium: Float
        meitnerium: Float
        darmstadtium: Float
        roentgenium: Float
        copernicium: Float
        nihonium: Float
        flerovium: Float
        moscovium: Float
        livermorium: Float
        tennessium: Float
        oganesson: Float
        comment: String
    }

    shiftstate{
        date: String
        time: String
        shiftid: String
        warninglevel: String
        remainingcartridges: Int
        remainingair: Int
        remainingelectricity: Int
        shift: ShiftMapper
        comment: String
    }

    sensordata{
        date: String
        time: String
        sourceid: String
        source: SensorMapper
        location: String
        event: String
        valuename: String
        value: Float
        units: String
    }

    sensor{
        id: String
        name: String
        location: LocationMapper
    }

    location{
        id: String
        name: String
        sensors: List
    }

    boat{
        id: String
        name: String
        capacity: Int
    }

= Examples =

http://localhost:1881/api/shiftstate/fields=shiftid,time,shift(id,start,end,chief(id))&where=minute:42

=== Mutations ===

= Entities =

create_location {
    name: String
}

remove_location {
    id: String
```

```
}

eradicate_location {
    id: String
}

create_sensor {
    location: String
    name: String
}

remove_sensor {
    id: String
}

create_department {
    name: String
    vk: String
}

remove_department {
    id: String
}

eradicate_department {
    id: String
}

create_specialization {
    name: String
}

remove_specialization {
    id: String
}

eradicate_ {
    id: String
}

create_person {
    department: String
    name: String
    patronymic: String
    phone: String
    specialization: String
    surname: String
}

remove_person {
    id: String
}

eradicate_person {
    id: String
}

create_boat {
    capacity: Int
    name: String
}

remove_boat {
    id: String
}

create_systemtype {
    description: String
    name: String
}

remove_systemtype {
    id: String
}

eradicate_systemtype {
    id: String
}

create_systemstate {
    description: String
```

```
    name: String
}

remove_systemstate {
    id: String
}

eradicate_systemstate {
    id: String
}

create_propertytype {
    description: String
    name: String
}

remove_propertytype {
    id: String
}

eradicate_propertytype {
    id: String
}

create_system {
    checked: String
    launched: String
    name: String
    serialnumber: Float
    state: String
    supervisor: String
    type: String
}

remove_system {
    id: String
}

create_property {
    admission: String
    comissioning: String
    department: String
    name: String
    type: String
}

remove_property {
    id: String
}

create_shift {
    chief: String
    department: String
    end: String
    requirements: String
    start: String
    workers: String
}

remove_shift {
    id: String
}

create_operation {
    end: String
    executors: String
    head: String
    name: String
    requirements: String
    start: String
}

remove_operation {
    id: String
}

create_requirement {
    content: String
    name: String
}
```

```
remove_requirement {
    id: String
}

create_position {
    attackangle: Float
    directionangle: Float
    speed: Float
    timestamp: String
    x: Float
    y: Float
    z: Float
}

remove_position {
    timestamp: String
}

create_controlaction {
    command: String
    mac: String
    params: String
    result: String
    timestamp: String
    user: String
}

remove_controlaction {
    timestamp: String
}

create_systemtest {
    result: Int
    system: String
    timestamp: String
}

remove_systemtest {
    timestamp: String
}

create_sensordata {
    event: String
    meaning: String
    source: String
    timestamp: String
    units: String
    value: Float
}

remove_sensordata {
    timestamp: String
}

create_shiftstate {
    comment: String
    remainingair: Int
    remainingcartridges: Int
    remainingelectricity: Int
    shift: String
    timestamp: String
    warninglevel: String
}

remove_shiftstate {
    timestamp: String
}

create_operationstate {
    actinium: Float
    aluminium: Float
    americium: Float
    argentum: Float
    argon: Float
    arsenicum: Float
    astatum: Float
    aurum: Float
    azimuth: Float
    barium: Float
    berkelium: Float
    beryllium: Float
```

bismuthum: **Float**
boat: String
bohrium: **Float**
borum: **Float**
bromum: **Float**
cadmium: **Float**
caesium: **Float**
calcium: **Float**
californium: **Float**
carboneum: **Float**
cerium: **Float**
chlorum: **Float**
chromium: **Float**
cobaltum: **Float**
comment: String
copernicium: **Float**
crypton: **Float**
cuprum: **Float**
curium: **Float**
darmstadtium: **Float**
distancetotheship: **Float**
dubnium: **Float**
dysprosium: **Float**
einsteinium: **Float**
erbium: **Float**
europium: **Float**
fermium: **Float**
ferrum: **Float**
flerovium: **Float**
fluorum: **Float**
francium: **Float**
gadolinium: **Float**
gallium: **Float**
germanium: **Float**
hafnium: **Float**
hassium: **Float**
helium: **Float**
holmium: **Float**
hydrargyrum: **Float**
hydrogenium: **Float**
indium: **Float**
iodium: **Float**
iridium: **Float**
kalium: **Float**
lanthanum: **Float**
lawrencium: **Float**
lithium: **Float**
livermorium: **Float**
lutetium: **Float**
magnesium: **Float**
manganum: **Float**
meitnerium: **Float**
mendelevium: **Float**
molybdaenum: **Float**
moscovium: **Float**
natrium: **Float**
neodymium: **Float**
neon: **Float**
neptunium: **Float**
niccolum: **Float**
nihonium: **Float**
niobium: **Float**
nitrogenium: **Float**
nobelium: **Float**
oganesson: **Float**
operation: String
osmium: **Float**
oxygenium: **Float**
palladium: **Float**
phosphorus: **Float**
platinum: **Float**
plumbum: **Float**
plutonium: **Float**
polonium: **Float**
praseodymium: **Float**
promethium: **Float**
protactinium: **Float**
radium: **Float**
radon: **Float**
rhenium: **Float**
rhodium: **Float**

```
    roentgenium: Float
    rubidium: Float
    ruthenium: Float
    rutherfordium: Float
    samarium: Float
    scandium: Float
    seaborgium: Float
    selenium: Float
    silicium: Float
    stannum: Float
    status: String
    stibium: Float
    strontium: Float
    sulfur: Float
    tantalum: Float
    technetium: Float
    tellurium: Float
    tennessium: Float
    terbium: Float
    thallium: Float
    thorium: Float
    thulium: Float
    timestamp: String
    titanium: Float
    uranium: Float
    vanadium: Float
    wolframium: Float
    xenon: Float
    ytterbium: Float
    yttrium: Float
    zenith: Float
    zincum: Float
    zirconium: Float
}

remove_operationstate {
    timestamp: String
}

= Examples =

http://localhost:1881/api/create/position/fields=ok,position(x)&where=timestamp:'2017-02-18 23:59:57',x:10.0,y:10.2,z:10
```

Logbook

В системе предусмотрена работа с содержимым базы данных при помощи API, предоставляемого разработчиком ODM cassandra driver. Однако, в соответствии с заданием была реализована возможность работы с БД на её "родном" языке.

Так, для выборки значений из базы данных можно воспользоваться функцией **select**, содержащейся в одноименном файле в папке native.

```
select('position', [['x', 10], ['y', 10], ['speed'], ['time']])
```

В том числе есть возможность исполнения скрипта с помощью командной строки

```
python3 native/select.py position x=10 y=10 speed time
python3 native/select.py position date='2018-04-03' time='20:43:03.426852000' speed
```

Формат результата

```
[{'date': Date(17624), 'time': Time(74583426852000), 'speed': 1488.0}]
```

Существует аналогичный скрипт для операции **update**

Пример вызова функции

```
update('position', [['date', '2018-04-03'], ['time', '20:43:03.426852000']], ['speed', 200]('speed', 200))
```

Пример вызова скрипта

```
python3 native/update.py position date='2018-04-03' time='20:43:03.426852000' 'speed->1488'
```

Указанные функции и скрипты предназначены для формирования запроса на языке cql и передачи его в СБД.

6.-Replication-and-sharding.md

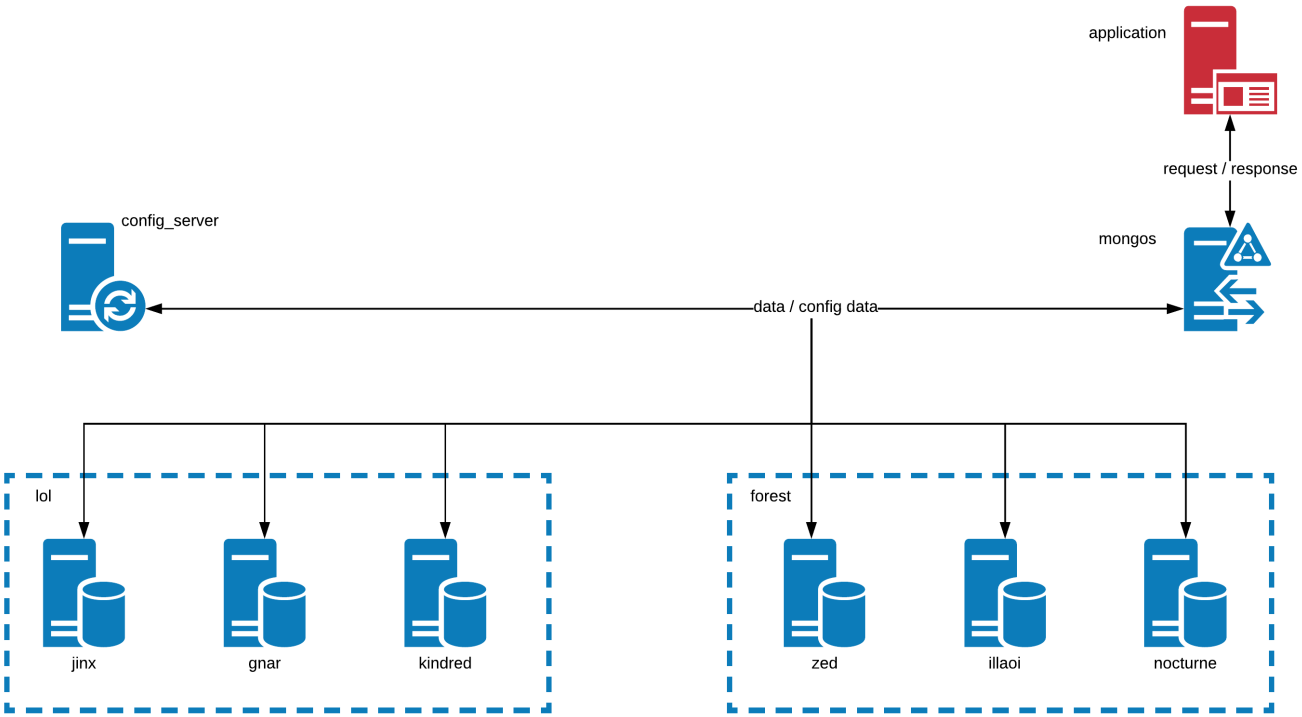
Common

Копии всех сконфигурированных серверов доступны здесь (https://cloud.mail.ru/public/7U1Y/TBDcQV1Yv) (для запуска использовалась ОС **Ubuntu 17.10**, версия mongo - **3.6.4**, cassandra - **3.0.16**, neo4j - **3.3.5**).

Mongo

Для базы данных recital, хранящейся в mongodb, в связи с репликацией необходимо запустить 8 процессов (используются соответственно порты 1882 - 1889):

View in full size (https://image.ibb.co/fJbOyy/mongo.png)



Три для replica set первого shard под названием 'lol' (путь к директориям, в которых хранятся файлы баз данных, указывается в соответствующих конфигурационных файлах)

```
mongod --config jinx_lol_shard_config.yaml
mongod --config gnar_lol_shard_config.yaml
mongod --config kindred_lol_shard_config.yaml
```

Три для replica set второго shard под названием 'forest'

```
mongod --config zed_forest_shard_config.yaml
mongod --config illaoi_forest_shard_config.yaml
mongod --config nocturne_forest_shard_config.yaml
```

Один в качестве сервера конфигурации mongos и replica sets

```
mongod --config config_server_config.yaml
```

Один в качестве драйвера (mongos)

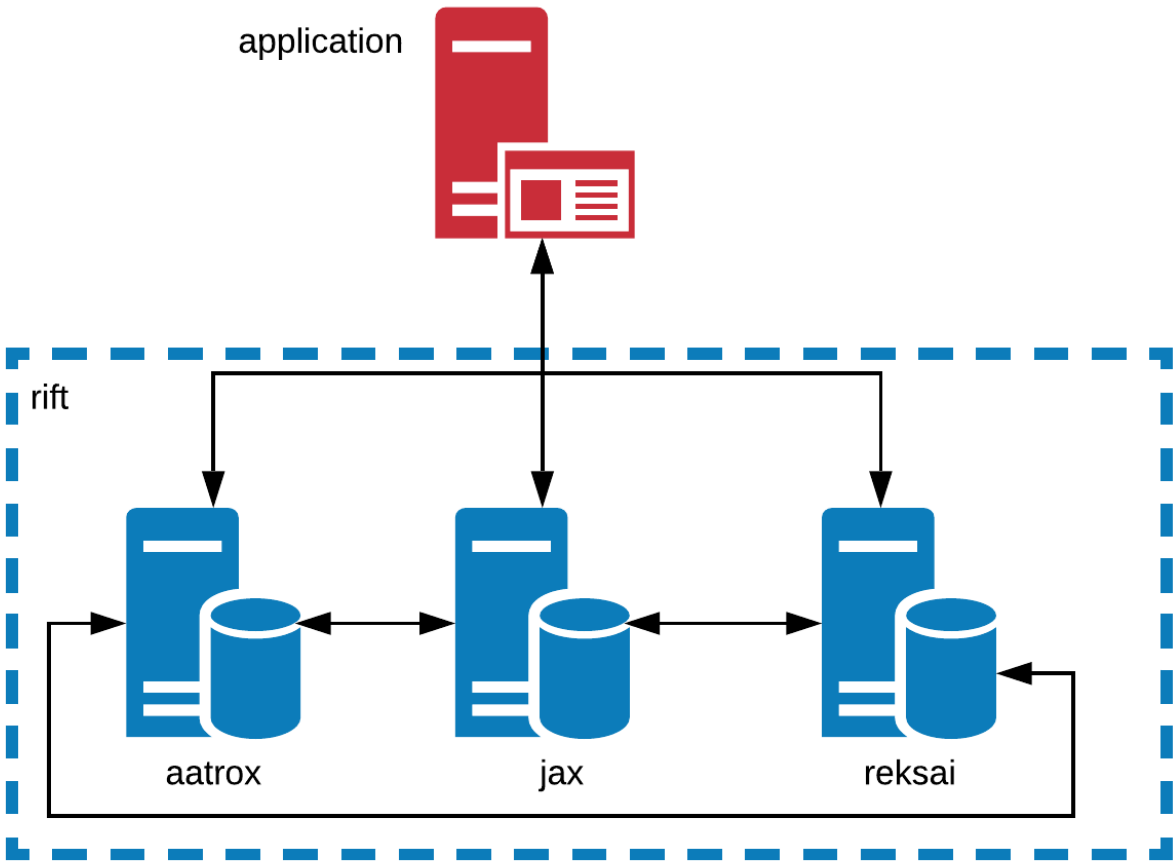
```
mongos --config mongos_config.yaml
```

Необходимо также обратить внимание на необходимость инициализации всех создаваемых replica sets. Соответствующие документы, который необходимо передавать в initiate - метод, приведены в json - файлах.

Cassandra

Делается как тут описано http://zcourts.com/2011/07/09/setting-up-a-multi-node-cassandra-cluster-on-a-single-windows-machine/#sthash.b3b1vCO7.8AmRM8mX.dpbs (http://zcourts.com/2011/07/09/setting-up-a-multi-node-cassandra-cluster-on-a-single-windows-machine/#sthash.b3b1vCO7.8AmRM8mX.dpbs) но с парой нюансов, а именно:

View in full size (https://image.ibb.co/crEXrJ/cassandra.png)



Добавить в файл hosts (`/etc/hosts`) алиасы для localhost

```
127.0.0.2   localhost
127.0.0.3   localhost
127.0.0.4   localhost
```

Создать папки для нодов кассандры

```
cassandra/
    ├── aatrox
    ├── jax
    └── reksai
```

Скачать и в каждую директорию распаковать download package

Например отсюда http://apache-mirror.rbc.ru/pub/apache/cassandra/3.0.16/apache-cassandra-3.0.16-bin.tar.gz (http://apache-mirror.rbc.ru/pub/apache/cassandra/3.0.16/apache-cassandra-3.0.16-bin.tar.gz)

Базовая конфигурация

Задать как минимум следующие поля для каждой ноды в файле `config/cassandra.yaml` : `cluster_name` , `seed_provider` - должны быть для всех одинаковы, `data_file_directories` , `commitlog_directory` , `saved_caches_directory` , `listen_address` , `rpc_address` - должны быть для всех разные.

Другие важные моменты конфигурации

1. Порт jmx меняется в файле `config/cassandra-env.sh` , он должен быть уникальным
2. Необходимо задать минимальный и минимальный размер памяти для jvm в файле `config/jvm.options` (например 100M), иначе java будет доходить до 2 гигов на каждой ноде.

Создание keyspace:

```
create keyspace if not exists logbook with replication = {'class' : 'SimpleStrategy', 'replication_factor' : 3};
```

Если все сделано правильно, то выполнение `describe keyspaces;` , выполненное на других репликах, должно содержать в том числе и только что созданное.

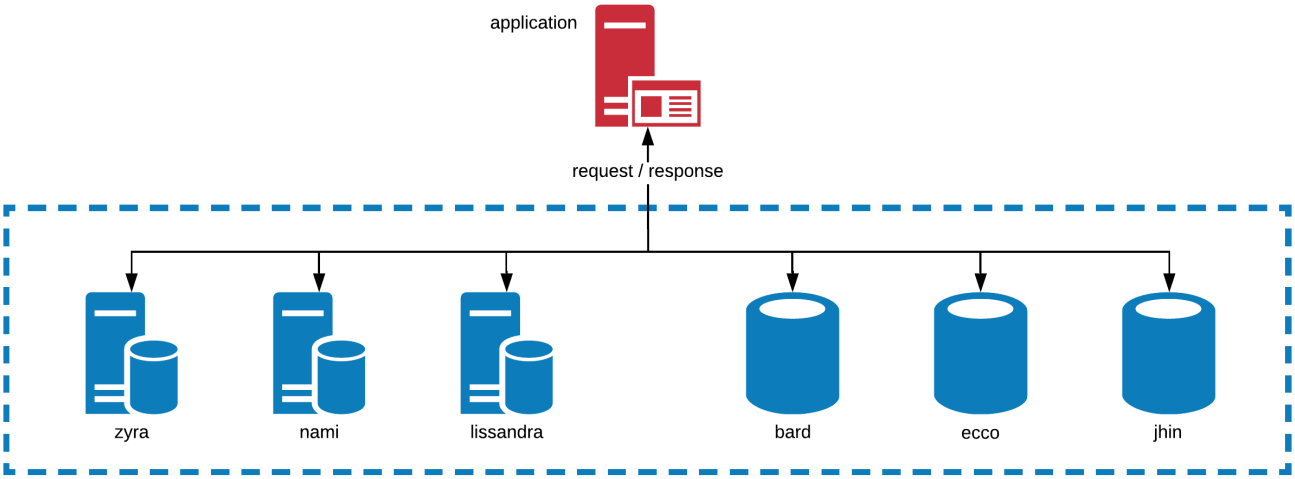
Далее можно сконфигурировать таблицы и заполнить их тестовыми данными.

Neo4j

Для развертывания кластера neo4j необходима enterprise edition

Делается как написано здесь https://neo4j.com/docs/operations-manual/current/tutorial/local-causal-cluster/ (https://neo4j.com/docs/operations-manual/current/tutorial/local-causal-cluster/). Далее будут приведены нюансы, характерные для данного приложения.

View in full size (https://image.ibb.co/jkgZMJ/neo4j.png)



Создать папки для нодов neo4j

```
neo4j/
    ├── bard
    ├── ekko
    ├── jhin
    ├── lissandra
    ├── nami
    └── zyra
```

Будем полагать, что `zyra` , `nami` , `lissandra` - core nodes, а `jhin` , `ekko` , `bard` - replicas.

To make backup:

```
./neo4j-admin backup
--backup-dir=/media/zeionara/8ef5f608-07b4-4216-88ce-1c255f8edaab/zeionara/neo4j/backups
--name=28.04.2018
--from=127.0.0.12:6363
--cc-report-dir=/media/zeionara/8ef5f608-07b4-4216-88ce-1c255f8edaab/zeionara/neo4j/backups
```

7.-Ilm.md

General description

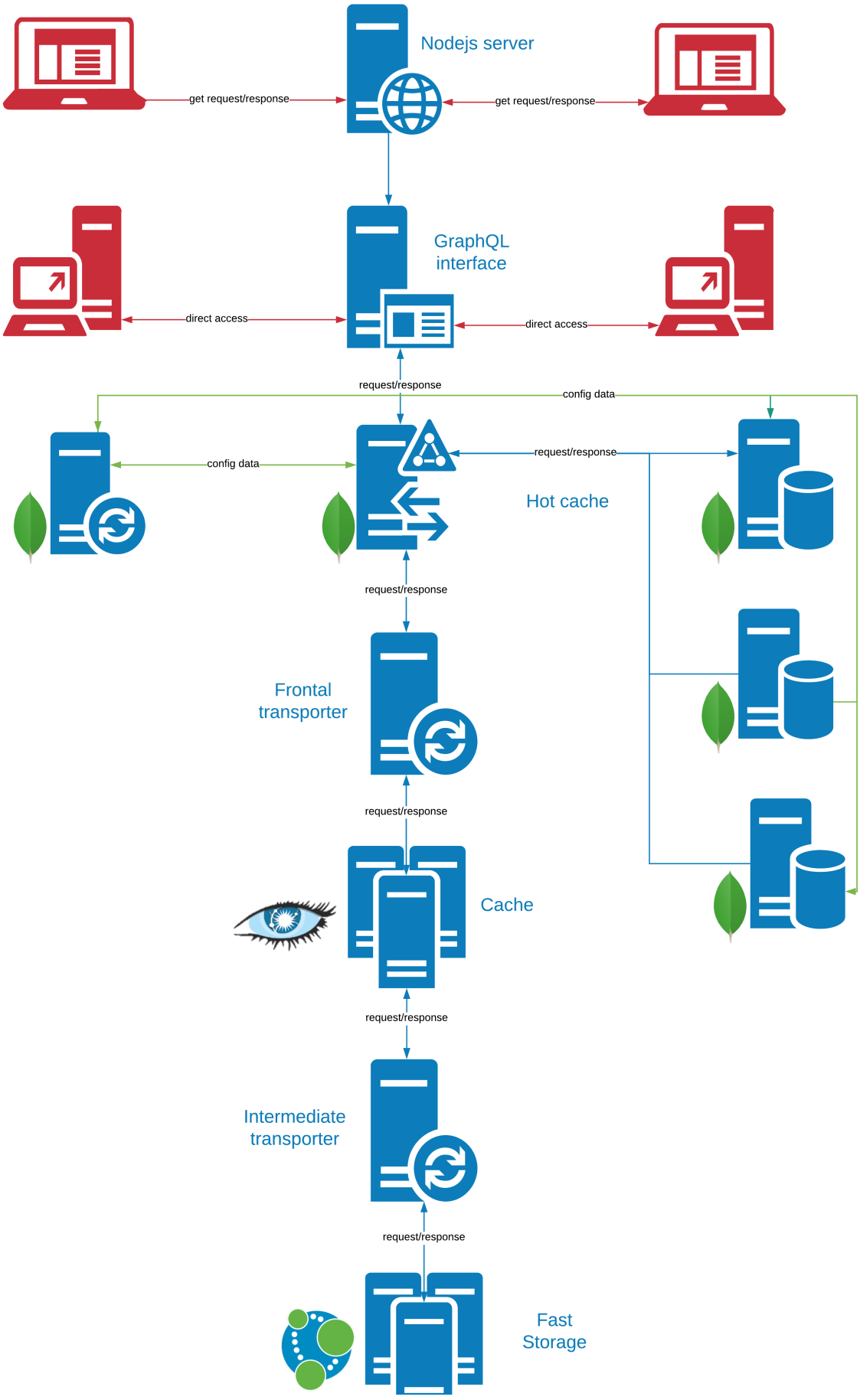
The ilm (which stands for **information lifecycle management**) model is a result of rethinking and restructuring the project (it is the main cause, why there is an standalone branch for the feature). So, in the old model the data kept in the database has been splitted up into three parts, which matches three dbms vendors depending on entity's functional dedication - some of them are located in mongo, the most basic data, collections, which represent journals and logs are located in cassandra in order to get statistics faster and connections between base entities are represented in neo4j, which gives a convenient way to visualize relations between objects.

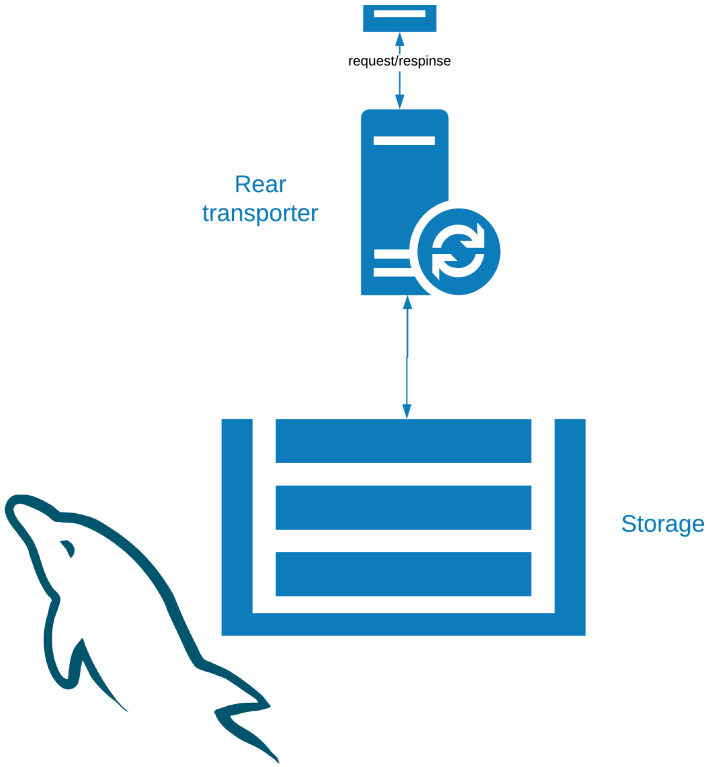
When building the ilm model an idea about keeping the most recent and the most needed data near to user in order to increase speed of fetching required data has been considered rather than functional dividing of data into several blocks. As a result each dbms from three vendors became a **layer** of the model, which might be primarily described by distance to the end-user. So, each dbms also came to containing schemas for every entity type

because now each entry might be occured on any layer of the model. Moreover, an additional external dbms has been taken to represent the lowest layer of the model. Because of low importance of required writing latency and speed as well as high requirements to the reliability, **RDBMS MySQL** has been used.

Architecture overview

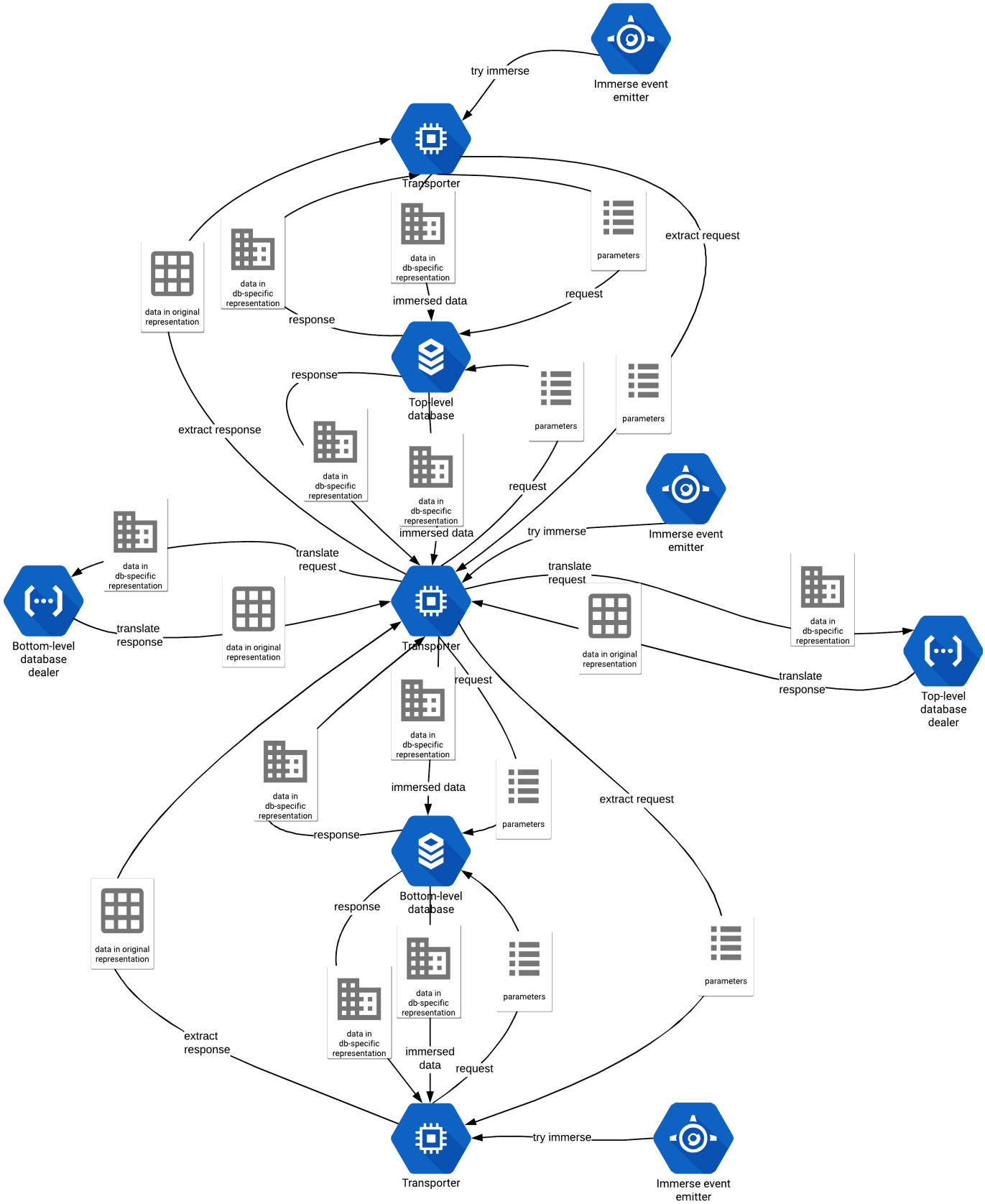
View in full size (https://image.ibb.co/hFFk17/Blank_Diagram_Page_1.png)





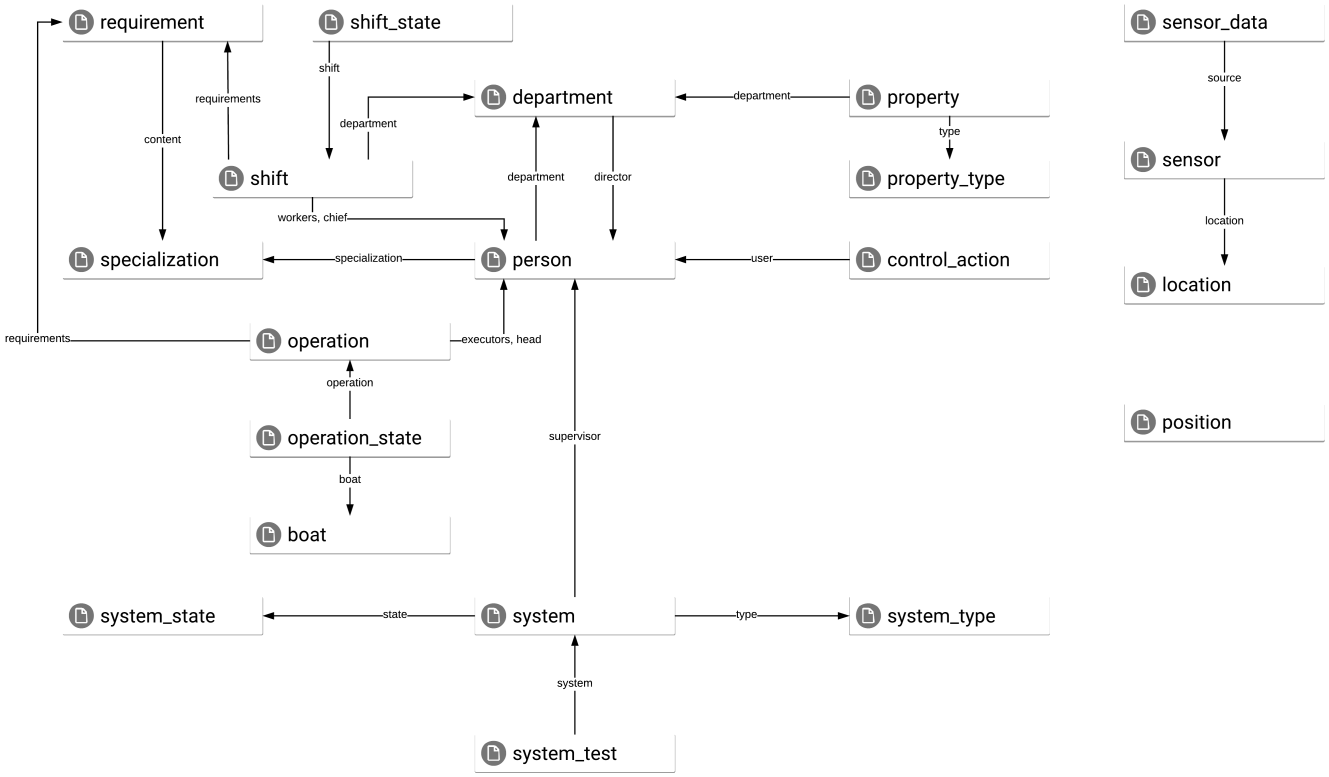
Transporter's interface overview

View in full size (https://image.ibb.co/gHsVrJ/System_of_Heroes_New_Page.png)



New ER model

View in full size (https://image.ibb.co/cW6owJ/er.png)



It might be noticed that now every object has a set of common fields - `_id`, `__accessed__`, `__created__`, `__gaps__` and `__cause__`.

`_id` represents a unique key, which can be used for addressing an entry as well as for building shard key (e.g. `ObjectId("000000000000000000000000c")`)

`__accessed__` keeps the moment, when the object have been accessed last time (e.g. `ISODate("2016-09-18T16:00:00Z")`).

`__created__` contains timestamp which points to the moment of entry creation (e.g. `ISODate("2016-09-18T16:00:00Z")`).

`__gaps__` contains distances (in seconds) between timestamps of accessing an entry (e.g. `[5, 60, 3600]` which means that the entry has been accessed at moments `__accessed__`, `__accessed__ - 3600`, `__accessed__ - 3660`, `__accessed__ - 3665`).

`__cause__` is a cause of moving an entry from upper level to a lower one (it is equal to `null` if an object has came from bottom in other cases it is a text value).

boats

property	data type
_id	blob
name	text
capacity	int
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

departments

property	data type
_id	blob
name	text
vk	text
director	blob

property	data type
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

property_types

property	data type
_id	blob
name	text
description	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

system_states

property	data type
_id	blob
name	text
description	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

system_types

property	data type
_id	blob
name	text
description	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

specializations

property	data type
_id	blob
name	text
__accessed__	timestamp
__created__	timestamp

property	data type
__gaps__	list of numbers
__cause__	text

locations

property	data type
_id	blob
name	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

sensors

property	data type
_id	blob
name	text
location	blob
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

systems

property	data type
_id	blob
name	text
type	blob
serial_number	float
launched	timestamp
checked	timestamp
supervisor	blob
state	blob
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

people

property	data type
_id	blob
name	text

property	data type
surname	text
patronymic	text
department	blob
phone	text
specialization	blob
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

properties

property	data type
_id	blob
name	text
type	blob
admission	timestamp
comissioning	timestamp
department	blob
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

shifts

property	data type
_id	blob
start	timestamp
end	timestamp
department	blob
chief	blob
workers	list of blob
requirements	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

operations

property	data type
_id	blob
name	text

property	data type
start	timestamp
end	timestamp
head	blob
executors	text
requirements	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

requirements

property	data type
_id	blob
name	text
content	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

system_tests

property	data type
_id	blob
timestamp	timestamp
system	blob
result	int
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

control_actions

property	data type
_id	blob
timestamp	timestamp
mac_address	blob
user	blob
command	text
params	text
result	text
__accessed__	timestamp

property	data type
__created__	timestamp
__gaps__	list of numbers
__cause__	text

positions

property	data type
_id	blob
timestamp	timestamp
x	float
y	float
z	float
speed	float
attack_angle	float
direction_angle	float
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

sensor_data

property	data type
_id	blob
timestamp	timestamp
source	blob
event	text
meaning	text
value	float
units	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

shift_states

property	data type
_id	blob
timestamp	timestamp
shift	blob
warning_level	text
_cartridges	int
air	int

property	data type
electricity	int
comment	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

operation_states

property	data type
_id	blob
timestamp	timestamp
boat	blob
operation	blob
status	text
distance	float
zenith	float
azimuth	float
hydrogenium	float
helium	float
lithium	float
beryllium	float
borum	float
carboneum	float
nitrogenium	float
oxygenium	float
fluorum	float
neon	float
natrium	float
magnesium	float
aluminium	float
silicium	float
phosphorus	float
sulfur	float
chlorum	float
argon	float
kalium	float
calcium	float
scandium	float
titanium	float
vanadium	float
chromium	float

property	data type
manganum	float
ferrum	float
cobaltum	float
niccolum	float
cuprum	float
zincum	float
gallium	float
germanium	float
arsenicum	float
selenium	float
bromum	float
crypton	float
_rubidium	float
strontium	float
yttrium	float
zirconium	float
niobium	float
molybdaenum	float
technetium	float
ruthenium	float
rhodium	float
palladium	float
argentum	float
cadmium	float
indium	float
stannum	float
stibium	float
tellurium	float
iodium	float
xenon	float
caesium	float
barium	float
lanthanum	float
cerium	float
praseodymium	float
neodymium	float
promethium	float
samarium	float
europium	float
gadolinium	float
terbium	float

property	data type
dysprosium	float
holmium	float
erbium	float
thulium	float
ytterbium	float
lutetium	float
hafnium	float
tantalum	float
wolframium	float
rhenium	float
osmium	float
_iridium	float
platinum	float
aurum	float
hydrargyrum	float
thallium	float
plumbum	float
bismuthum	float
polonium	float
astatum	float
radon	float
francium	float
radium	float
actinium	float
thorium	float
protactinium	float
uranium	float
neptunium	float
plutonium	float
americium	float
curium	float
berkelium	float
californium	float
einsteinium	float
fermium	float
mendelevium	float
nobelium	float
lawrencium	float
rutherfordium	float
dubnium	float
seaborgium	float

property	data type
bohrium	float
hassium	float
meitnerium	float
darmstadtium	float
roentgenium	float
copernicium	float
nihonium	float
flerovium	float
moscovium	float
livermorium	float
tennessium	float
oganesson	float
comment	text
__accessed__	timestamp
__created__	timestamp
__gaps__	list of numbers
__cause__	text

Заключение

Итак, в результате выполнения лабораторной работы было произведено ознакомление с основными ключевыми типами баз данных, которые концептуально отличаются от реляционных, хранящих данные в виде взаимосвязанных таблиц.

Были сделаны выводы об области применения отдельных типов баз данных в производственной деятельности, а именно, документо-ориентированная модель наиболее универсальная и обладает гораздо большей гибкостью по сравнению с общепринятыми SQL базами данных. Колоночная база данных в высшей степени предназначена для ведения разного рода логов и журналов, с целью дальнейшей группировки данных по столбцам при помощи пользовательских или встроенных агрегирующих фунций, именно такие базы данных ориентированы на режимы работы типа OLAP. Третий тип баз данных, графовые, выделяются наибольшей наглядностью и простотой построения моделей данных, что роднит принципы организации таких баз данных с объектно-ориентированной парадигмой разработки ПО.

Однако, как было показано во второй половине курсовой работы, все типы баз данных могут хранить в себе любые данные вне зависимости от области их применения, хотя в таком случае показатели быстродействия и удобства применения могут оказаться гораздо ниже.

Также выполнение курсовой работы способствовало ознакомлению с программными продуктами, упрощающими взаимодействие с базами данных при помощи таких языков программирования, как Java и Python. Была на практике реализована возможность создания пользовательских функций, которые могли бы быть вызваны тем же способом, что и встроенные.

Помимо всего прочего, были написаны скрипты для генерации тестовых данных при помощи сторонней библиотеки. Было измерено и оценено время заполнения базы данных большими объемами данных, которое, как оказалось, сильно зависит от используемой технологии.