**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ**

**ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»**

Институт компьютерных наук и технологического образования

Кафедра компьютерных технологий и электронного обучения

**ОТЧЕТ**

ПРОЕКТ БАЗЫ ДАННЫХ ДЛЯ САЙТА ПО ПОИСКУ РАБОТЫ ДЛЯ СТУДЕНТОВ И ВЫПУСКНИКОВ НА САЙТЕ ВУЗА

Направление подготовки: «Информатика и вычислительная техника»

Руководитель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Жуков Н.Н.

« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Авторы работы студенты

2 Группы ИВТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.В. Иванов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.C. Ханов

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.А. Храмов

« \_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

Оглавление

[ПРЕДМЕТНАЯ ОБЛАСТЬ 3](#_Toc11836112)

[Почему MySQL? 3](#_Toc11836113)

[Почему не NoSQL? 4](#_Toc11836114)

[Нормализация 5](#_Toc11836115)

[UML ER-диаграмма 6](#_Toc11836116)

[Исходный текст запросов 7](#_Toc11836118)

## ПРЕДМЕТНАЯ ОБЛАСТЬ

В своей жизни мы принимаем ряд важных, подчас судьбоносных решений. Выбор места работы – одно из них, ведь от того, в какой организации, должности, окружении работает человек, зависит его социальный статус, материальное благополучие, самочувствие и самоощущение. Именно с поиска достойной работы начинается или успешно продолжается деловая карьера. Известный афоризм "Кто ищет, тот всегда найдет!" как нельзя лучше подходит к поиску работы на современном рынке труда.

Но также активным поиском занимаются многочисленные компании и проектные группы. Уже на первых этапах проектирования базы данных для поиска работы, необходимо учитывать потребности не только соискателей, но и работодателей.

Помимо двух основных групп пользователей на сайте университета присутствует команда операторов, в чьи обязанности входит прием и проверка всех объявлений на сайте.

Чтобы не только учесть потребности трех групп пользователей (ведь операторы тоже в определенном смысле ими являются), но и облегчить внедрение нового функционала, мы совместно с научным руководителем выявили основные элементы базы данных. Среди них имеются идентификаторы: операторов, пользователей, компаний, филиалов университета, резюме, вакансий, областей и городов соискателей и филиалов.

## Почему MySQL?

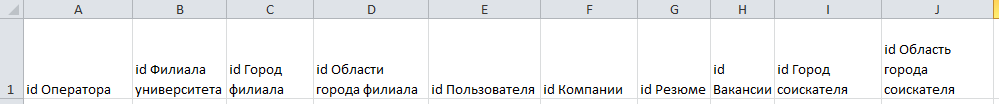
* Нет необходимости вникать в тонкости настроек
* В любом языке программирования, фреймворке и так далее и тому подобное — есть интеграция с MySQL
* Высокий порог вхождения
* MySQL поддерживает большую часть функционала SQL.
* Безопасность:в MySQL встроено много функций безопасности.
* Мощность и масштабируемость:MySQL может работать с действительно большими объёмами данных, и неплохо походит для масштабируемых приложений.
* Скорость:пренебрежение некоторыми стандартами позволяет MySQL работать производительнее, местами “срезая на поворотах”.

## Почему не NoSQL?

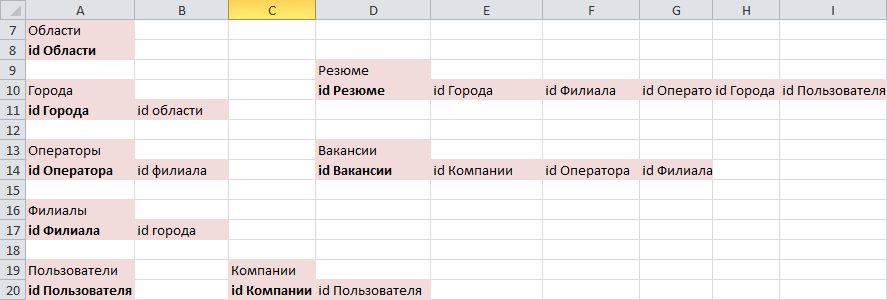
Для того, чтобы прийти к простому и понятному выводу, давайте проанализируем разницу между SQL- и NoSQL-подходами:

* Структура и тип хранящихся данных: SQL/реляционные базы данных требуют наличия однозначно определённой структуры хранения данных, а NoSQL базы данных таких ограничений не ставят.
* Запросы: вне зависимости от лицензии, РСУБД реализуют SQL-стандарты, поэтому из них можно получать данные при помощи языка SQL. Каждая NoSQL база данных реализует свой способ работы с данными.
* Масштабируемость: оба решения легко растягиваются вертикально (например, путём увеличения системных ресурсов). Тем не менее, из-за своей современности, решения NoSQL обычно предоставляют более простые способы горизонтального масштабирования (например, создания кластера из нескольких машин).
* Надёжность: когда речь заходит о надёжности, SQL базы данных однозначно впереди.
* Поддержка: РСУБД имеют очень долгую историю. Они очень популярны, и поэтому получить поддержку, платную или нет, очень легко. Поэтому, при необходимости, решить проблемы с ними гораздо проще, чем с NoSQL, особенно если проблема сложна по своей природе (например, при работе с MongoDB).
* Хранение и доступ к сложным структурам данных: по своей природе реляционные базы данных предполагают работу с сложными ситуациями, поэтому и здесь они превосходят NoSQL-решения.

## Нормализация

Первая нормальная форма

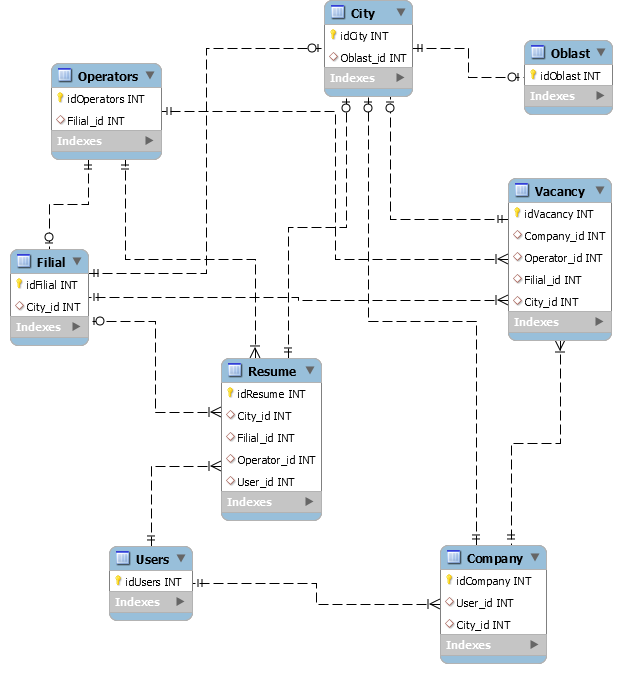
Третья нормальная форма



* каждый неключевой атрибут неприводимо зависит от (каждого) её потенциального ключа;
* отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых;

## UML ER-диаграмма

## 



## Исходный текст запросов

-- -----------------------------------------------------

-- Schema Work

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Work` ;

USE `Work` ;

-- -----------------------------------------------------

-- Table `Work`.`Oblast`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Oblast` (

`idOblast` INT UNSIGNED NOT NULL,

PRIMARY KEY (`idOblast`),

UNIQUE INDEX `idOblast\_UNIQUE` (`idOblast` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`City`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`City` (

`idCity` INT UNSIGNED NOT NULL AUTO\_INCREMENT,

`Oblast\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idCity`),

INDEX `Oblast\_id\_idx` (`Oblast\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Oblast\_id`

FOREIGN KEY (`Oblast\_id`)

REFERENCES `Work`.`Oblast` (`idOblast`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Users` (

`idUsers` INT UNSIGNED NOT NULL,

PRIMARY KEY (`idUsers`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Company`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Company` (

`idCompany` INT UNSIGNED NOT NULL,

`User\_id` INT UNSIGNED NULL,

`City\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idCompany`),

INDEX `User\_id\_idx` (`User\_id` ASC) VISIBLE,

INDEX `fk3\_City\_id\_idx` (`City\_id` ASC) VISIBLE,

CONSTRAINT `fk\_User\_id`

FOREIGN KEY (`User\_id`)

REFERENCES `Work`.`Users` (`idUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk4\_City\_id`

FOREIGN KEY (`City\_id`)

REFERENCES `Work`.`City` (`idCity`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Filial`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Filial` (

`idFilial` INT UNSIGNED NOT NULL,

`City\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idFilial`),

INDEX `fk\_City\_id\_idx` (`City\_id` ASC) VISIBLE,

CONSTRAINT `fk\_City\_id`

FOREIGN KEY (`City\_id`)

REFERENCES `Work`.`City` (`idCity`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Operators`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Operators` (

`idOperators` INT UNSIGNED NOT NULL,

`Filial\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idOperators`),

INDEX `fk\_Filial\_id\_idx` (`Filial\_id` ASC) VISIBLE,

CONSTRAINT `fk\_Filial\_id`

FOREIGN KEY (`Filial\_id`)

REFERENCES `Work`.`Filial` (`idFilial`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Vacancy`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Vacancy` (

`idVacancy` INT NOT NULL,

`Company\_id` INT UNSIGNED NULL,

`Operator\_id` INT UNSIGNED NULL,

`Filial\_id` INT UNSIGNED NULL,

`City\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idVacancy`),

INDEX `Company\_id\_idx` (`Company\_id` ASC) VISIBLE,

INDEX `fk\_Operator\_id\_idx` (`Operator\_id` ASC) VISIBLE,

INDEX `fk\_Filial\_id\_idx` (`Filial\_id` ASC) VISIBLE,

INDEX `fk2\_City\_id\_idx` (`City\_id` ASC) VISIBLE,

CONSTRAINT `fk2\_Company\_id`

FOREIGN KEY (`Company\_id`)

REFERENCES `Work`.`Company` (`idCompany`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk2\_Operator\_id`

FOREIGN KEY (`Operator\_id`)

REFERENCES `Work`.`Operators` (`idOperators`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk2\_Filial\_id`

FOREIGN KEY (`Filial\_id`)

REFERENCES `Work`.`Filial` (`idFilial`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk2\_City\_id`

FOREIGN KEY (`City\_id`)

REFERENCES `Work`.`City` (`idCity`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Work`.`Resume`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Work`.`Resume` (

`idResume` INT NOT NULL,

`City\_id` INT UNSIGNED NULL,

`Filial\_id` INT UNSIGNED NULL,

`Operator\_id` INT UNSIGNED NULL,

`User\_id` INT UNSIGNED NULL,

PRIMARY KEY (`idResume`),

INDEX `fk\_City\_id\_idx` (`City\_id` ASC) VISIBLE,

INDEX `fk\_Filial\_id\_idx` (`Filial\_id` ASC) VISIBLE,

INDEX `fk\_Operator\_id\_idx` (`Operator\_id` ASC) VISIBLE,

INDEX `fk\_User\_id\_idx` (`User\_id` ASC) VISIBLE,

CONSTRAINT `fk1\_City\_id`

FOREIGN KEY (`City\_id`)

REFERENCES `Work`.`City` (`idCity`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk1\_Filial\_id`

FOREIGN KEY (`Filial\_id`)

REFERENCES `Work`.`Filial` (`idFilial`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk1\_Operator\_id`

FOREIGN KEY (`Operator\_id`)

REFERENCES `Work`.`Operators` (`idOperators`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk1\_User\_id`

FOREIGN KEY (`User\_id`)

REFERENCES `Work`.`Users` (`idUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;