**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Курсовая работа

Использование NoSQL баз данных

Выполнили Мохнаткин Д.А.

Шумеев А.А.

Группа P3318

2017 г.

# Цель курсовой работы

Получение навыков и знаний, необходимых для создания баз данных, основанной на

принципах, отличных от классической реляционной (и объектно-реляционной) модели, в

произвольной предметной области. Применение этих навыков и знаний для реализации БД

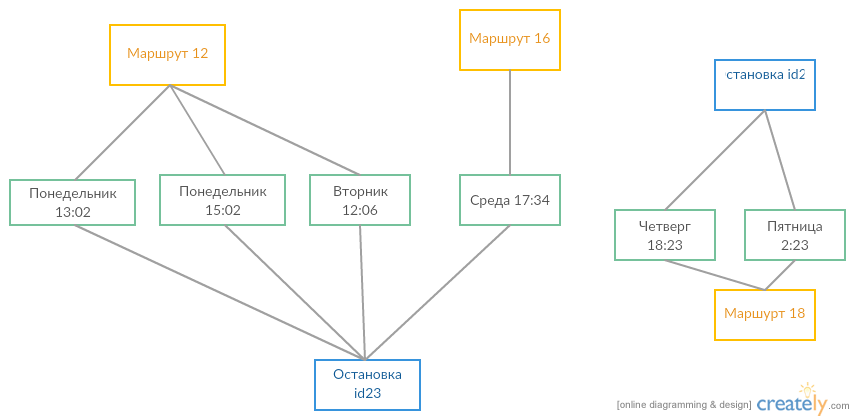
по выбранной исполнителями предметной области.

# Описание предметной области

В курсовой работе будет реализовано хранение данных для системы контроля общественного транспорта некоторого города. Т.к. таких данных может быть много, и они разнотипные, будет использовано несколько разных баз данных (разных типов).

Разделим данные на 4 группы:

1. **Данные о транспортном средстве.** Подразумевается возможность хранить много вспомогательной информации для каждого тс. Например, номер, дата покупки, дата выпуска, дата прохождения тех. осмотра и т.д. Невозможно предусмотреть все возможные характеристики, поэтому будем использовать документно-ориентированную СУБД. (MongoDb)
2. **Данные о местоположении транспортного средства.** Для каждого транспортного средства будет отслеживаться его текущее местоположение. Т.к. это местоположение будет часто обновляться (раз в несколько секунд), то требуется использовать быстрое хранилище. Например, Redis.
3. **Данные об остановках общественного транспорта.** Для каждой остановки требуется хранить ее id, географическое местоположение (в виде координат) и название. Для обеспечения быстродействия запросов типа: найти остановки в некотором районе, или найти ближайшую остановку, или найти остановку по имени и т.д. будем использовать колоночную СУБД (Сassandra).
4. **Маршруты общественного транспорта.** Удобно хранить в виде графа с 3-мя типами нод (нода с id остановки, нода с временем, нода с id маршрута). Связями обозначать когда на остановку должен приезжать автобус с определенным номером маршрута. (Neo4j).

****

1. **Данные о персонале.** Перечень сотрудников (водители, уборщицы, механики, и т.д.) а также на какие маршруты назначены водители. (MySql)

# Описание программных модулей в формате комментариев к коду

**Данные о транспортном средстве.**

[Produces("application/json")]

[Microsoft.AspNetCore.Mvc.Route("api/units")]

public class UnitsController : Controller {

// GET: api/Units/5

[HttpGet("{id}")]

public IActionResult Get(int id) {

return Ok(GetUnitsCollection().Find(x => x.Id == id).FirstOrDefault());

}

**Данные о местоположении транспортного средства.**

[Produces("application/json")]

[Route("api/unit")]

// Весь api о данных местоположения транспортного средства в этом контроллере

public class UnitPositionController : Controller {

private readonly IRedisClientsManager \_redisManager;

public UnitPositionController(IRedisClientsManager redisManager){

\_redisManager = redisManager;

}

// GET api/unit/5

[HttpGet("{id}")]

public Position Get(int id) {

using (var redis = \_redisManager.GetClient()){

return redis.ContainsKey(id.ToString()) ?

Position.Parse(redis.Get<string>(id.ToString())) : null;

}

}

**Данные об остановках общественного транспорта.**

[Produces("application/json")]

[Route("api/UnitStops")]

// Весь api об остановках общественного транспорта

public class UnitStopsController : Controller {

private readonly ICluster \_cluster;

public const string KeySpace = "db";

public UnitStopsController(ICluster cluster){

\_cluster = cluster;

}

// GET: api/UnitStops/5

[HttpGet("{id}")]

public UnitStop Get(int id){

using (var session = \_cluster.Connect(KeySpace)){

Row res = session.Execute($"select \* from bus\_stops where id = {id};").FirstOrDefault();

if (res == null)

return null;

return new UnitStop{

Id = id,

Title = res.GetValue<string>("title"),

Latitude = res.GetValue<double>("latitude"),

Longitude = res.GetValue<double>("longitude")

};

}

}

**Маршруты общественного транспорта.**

[Produces("application/json")]

[Route("api/routes")]

// Весь api о данных об остановках общественного транспорта

public class RoutesController : Controller {

private IGraphClient \_graphClient;

public RoutesController(IGraphClient graphClient){

\_graphClient = graphClient;

}

/// <summary>

/// Delete relation between bus stop and route number

/// </summary>

[HttpDelete]

public IActionResult DeleteScheduleEntry([FromBody]ScheduleEntry scheduleEntry){

\_graphClient.Cypher

.Match("(route:Route)-[:STOPS\_AT\_TIME]->(timing:Timing)-[:APPLIES\_TO\_STOP]->(stop:Stop)")

.Where((RouteNeo4JModel route, UnitStopNeo4JModel stop) =>

route.id == scheduleEntry.RouteNumber && stop.id == scheduleEntry.UnitStopId)

.Delete("timing")

.ExecuteWithoutResults();

return Ok();

}

**Данные о персонале.**

[Produces("application/json")]

[Route("api/Staff")]

// Весь api о данных о персонале

public class StaffController : Controller {

private IStaffDriver \_staffDriver;

public StaffController(IStaffDriver staffDriver) {

\_staffDriver = staffDriver;

}

// GET: api/Staff/5

[HttpGet("{id}", Name = "Get")]

public IActionResult Get(int id) {

var result = \_staffDriver.Get(id);

if (result != null)

return Ok(result);

return NotFound();

}