NoSQL Модель данных

1. Графическое представление

Графическое представление модели данных Neo4j показана на рис. 1

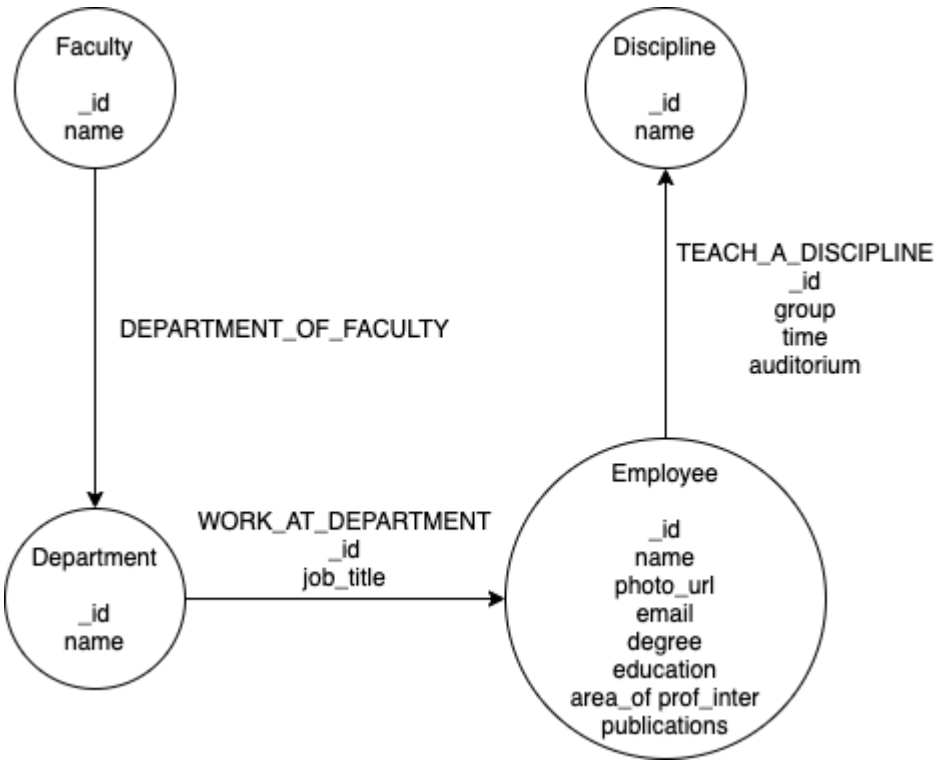

Рисунок 1 - Графическое представление модели данных Neo4j

2. Описание назначений коллекций, типов данных и сущностей

Модель состоит из 4 сущностей:

1. Факультет

- “_id”, int - идентификатор. 4B
- “name”, String - название факультета 50*2B

Итого: 104B

2. Кафедра

- “_id”, int - идентификатор. 4B
- “name”, String - название кафедры 50*2B

Итого: 104B

3. Работник

- “_id”, int - идентификатор. 4B
- “name”, String - ФИО 20*2B

- “photo_url”, String - URL фотографии 50*2B
- “email”, String - Email 30*2B
- “degree”, String - Научная степень 10*2B
- “education”, String – Образование 50*2B
- “area_of_prof_inter”, String – Область профессиональных интересов 500*2B
- “publications”, String – Публикации в научных изданиях 500*2B

Итого: 2324B

4. Дисциплина

- “_id”, int - идентификатор. 4B
- “name”, String - название дисциплины 20*2B

Итого: 44B

Существуют 4 связи между сущностями:

1. DEPARTMENT_OF_FACULTY - Факультет>Кафедра. Данный факультет содержит данные кафедры. Дополнительных полей нет.

2. WORK_AT_DEPARTMENT - Кафедра>Работник. На данной кафедре работают данные сотрудники. Дополнительные поля:

- “_id”, int - идентификатор. 4B
- “job_title”, String - наименование должности 30*2B

Итого: 64B

3. TEACH_A_DISCIPLINE - Работник>Дисциплина. Работник преподает данную дисциплину в указанное время и в указанной аудитории. Дополнительные поля:

- “_id”, int - идентификатор. 4B
- “group”, int - группа 4B
- "time”, String - время занятия 20*2B
- “auditorium”, String- аудитория 6*2B

Итого: 60B

4. Расчет объема

Предположим, что имеется F факультета, на каждом факультете по K кафедры, на каждой кафедре по E преподавателей, а каждый преподаватель преподает по D дисциплины и ведет по P занятия в неделю.

Чистый объем:

- “Факультет” - 104B
- “Кафедра” - 104B,
- “Работник” - 2324B
- “Дисциплина” - 44B
- WORK_AT_DEPARTMENT (W) – 64B
- TEACH_A_DISCIPLINE (T) – 60B.

Тогда занимаемый “чистый” объем: (104B*F + 104B*K + 2324B*E + 44B*D + 64B*W + 60B*T)

Фактический объем:

- “Факультет” - 248B
- “Кафедра” - 248B
- “Работник” - 3272B
- “Дисциплина” - 224B
- DEPARTMENT_OF_FACULTY (G) - 34B
- WORK_AT_DEPARTMENT (W) – 263B
- TEACH_A_DISCIPLINE (T) – 424B

Тогда фактический объем БД: (248B*F + 248B*K + 3272B*E + 224B*D + 263B*W + 424B*T + 34B*G)

Избыточность модели: (248B*F + 248B*K + 3272B*E + 224B*D + 263B*W + 424B*T + 34B*G) / (104B*F + 104B*K + 2324B*E + 44B*D + 64B*W + 60B*T)

3. Примеры запросов

1. Добавить сотрудника

   CREATE (e:Employee {...})

2. Найти сотрудников, которые работают на кафедре с id = 2

   MATCH (k {_id: 2})-[:WORK_AT_DEPARTMENT]-(e)
   RETURN e.name, …..

SQL Модель данных

1. Графическое представление

Графическое представление модели данных MySQL показана на рис. 2

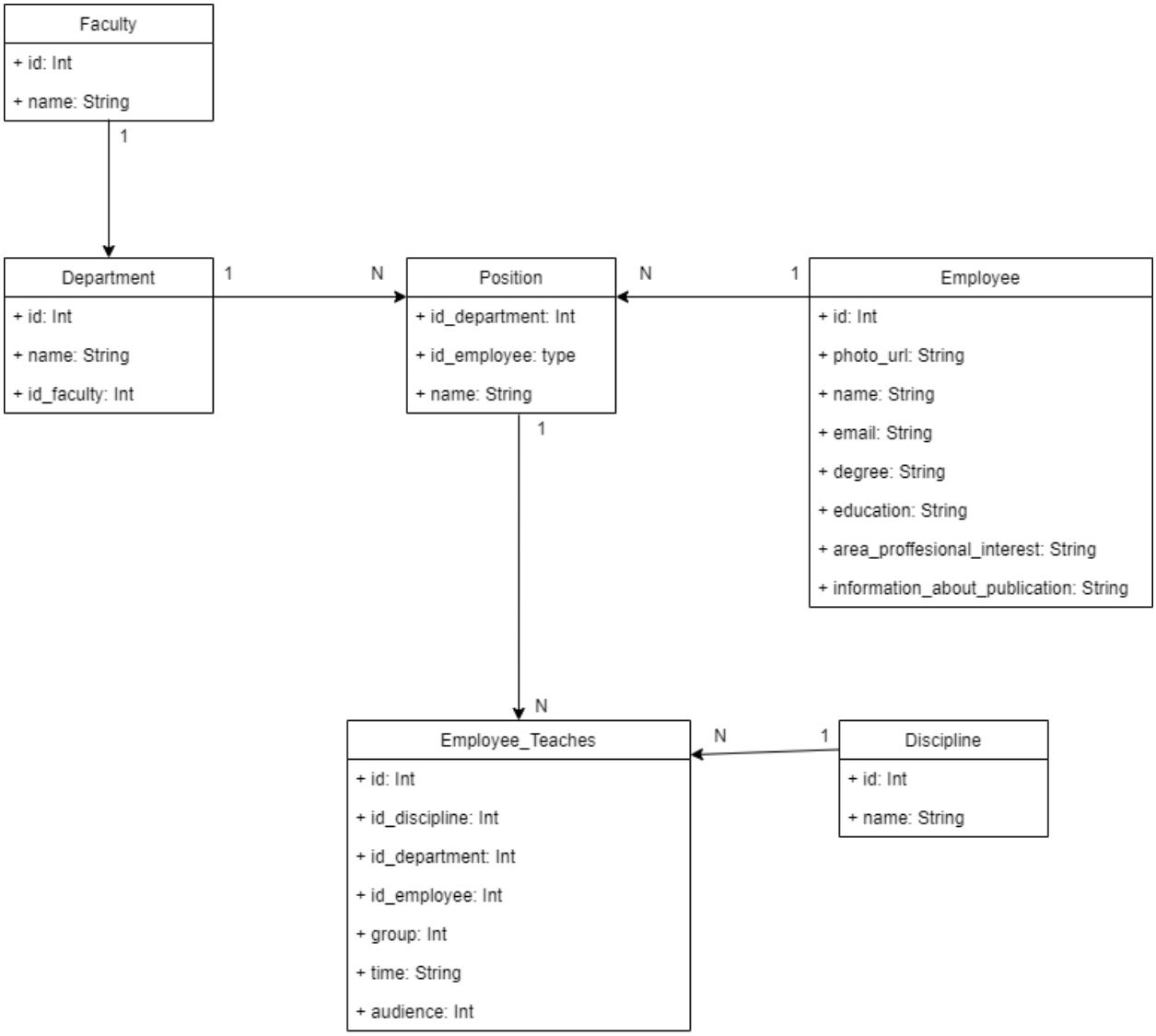

Рисунок 2 - Графическое представление модели данных MySQL

2. Описание назначений коллекций, типов данных и сущностей

В качестве реляционной СУБД использована MySql, в которой создано 6 таблицы: Faculty – хранит информацию о факультетах, Department – хранит информацию о кафедре, Position – соединяет сотрудника и его должность на кафедре, Employee – хранит информацию о сотруднике, Discipline – хранит информацию о дисциплине, Employee_Teaches – хранит информацию о преподаваемых дисциплинах сотрудником на кафедре.

Таблица Faculty содержит следующие поля:

- id – уникальный идентификатор факультета. Тип – Int. V = 4B
- name – название факультета. Тип – String. V = 50 * 2B = 100B

Итого: V = 104B

Таблица Department содержит следующие поля:

- id – уникальный идентификатор кафедры. Тип – Int. V = 4B
- name – название кафедры. Тип – String. V = 50 * 2B = 100B
- id_faculty – ссылка на фикультет. Тип – Int. V = 4B

Итого: V = 108B

Таблица Employee содержит следующие поля:

- id – уникальный идентификатор сотрудника. Тип – Int. V = 4B
- photo_url – ссылка на фотографию. Тип – String. V = 50 * 2B = 100B
- name – название факультета. Тип – String. V = 20 * 2B = 40B
- email – почта сотрудника. Тип – String. V = 30 * 2B = 60B
- degree – ученная степень. Тип – String. V = 10 * 2B = 20B
- education – образование. Тип – String. V = 50 * 2B = 100B
- area_proffesional_interest – область профессиональных интересов. Тип – String. V = 500 * 2B = 1000B
- information_about_publication – информацию о публикациях. Тип – String. V = 500 * 2B = 1000B

Итого: V = 2324B

Таблица Position имеет следующие поля:

- id_department – ссылка на кафедру. Тип – Int. V = 4B
- id_employee – ссылка на сотрудника. Тип – Int. V = 4B
- name – название должности. Тип – String. V = 30 * 2B = 60B

Итого: V = 68B

Таблица Discipline имеет следующие поля:

- id – уникальный идентификатор дисциплины. Тип – Int. V = 4B
- name – название дисциплины. Тип – String. V = 20 * 2B = 40B

Итого: V = 44B

Таблица Employee_Teaches имеет следующие поля:

- id – уникальный идентификатор занятий. Тип – Int. V = 4B
- id_discipline – ссылка на дисциплину. Тип – Int. V = 4B
- id_department – ссылка на кафедру. Тип – Int. V = 4B
- id_employee – ссылка на сотрудника. Тип – Int. V = 4B
- group – номер группы. Тип – Int. V = 4B
- time – время занятия. Тип – String. V = 20*2B = 40B
- audience – номер аудитории. Тип – String. V = 6*2B = 12B

Итого: V = 72B

4. Расчет объема

Предположим, что имеется F факультета, на каждом факультете по K кафедры, на каждой кафедре по E преподавателей, а каждый преподаватель преподает по D дисциплины и ведет по P занятия в неделю.

Чистый объем:

- F*104B – количество факультетов
- F*K*108B – количество кафедр
- E*2324B – количество сотрудников
- F*K*E*2324B – количество должностей
- D*44 – количество дисциплин
- F*K*E*D*P*72B – количество занятий

Чистый объем БД = F*104 + F*K*108 + E*2324 * F*K*E*2324 + D*44 * F*K*E*D*P*72

Фактически объем:

- F*104B – количество факультетов
- F*K*112B – количество кафедр
- E*2324B – количество сотрудников
- F*K*E*2332B – количество должностей
- D*44 – количество дисциплин
- F*K*E*D*P*80B – количество занятий

Фактический объем БД = F*104 + F*K*112 + E*2324 * F*K*E*2332 + D*44 * F*K*E*D*P*80

Избыточность модели: (F*104 + F*K*112 + E*2324 * F*K*E*2332 + D*44 * F*K*E*D*P*80)/(F*104 + F*K*112 + E*2324 * F*K*E*2332 + D*44 * F*K*E*D*P*80)

3. Примеры запросов

1. Добавить сотрудника

INSERT INTO Employee VALUES (…)

INSERT INTO Position VALUES (…)

2. Найти сотрудников, которые работают на кафедре с id = 2

SELECT * FROM Employee INNER JOIN Position ON Position.id_employee = Employee.Id AND Position.id_department = 2

Сравнение Neo4j и SQL модели данных

Запросы на SQL – языке являются более громоздкими, чем на языке Neo4j Cypher. Реализация модели данных в MySQL потребовало бы создать более количество сущностей, чтобы связывать данные между собой.

Neo4j требует больше памяти по сравнение с MySQL для хранения данных, но время выполнения запросов в Neo4j в среднем составляет меньше, чем в MySQL (Neo4j – 57 мс, MySQL – 108 мс)

Исходя из вышеперечисленного, можно сказать, что для рассматриваемой задачи Neo4j подходит в большей мере.