Московский авиационный институт (национальный исследовательский университет)
Институт № 8 «Компьютерные науки и прикладная математика»
Кафедра № 806 «Вычислительная математика и программирование»

Проектирование системы переноса и генерации взаимосвязанных данных из производственной среды при тестировании образовательной платформы

Выпускная квалификационная работа бакалавра

Студент группы М8О-406Б-21: Мезенин Олег Александрович
Научный руководитель: ст. преподаватель кафедры 806
Миронов Евгений Сергеевич

Москва — 2025

Актуальность темы

- Тестирование программного продукта важно.
- Для тестовых сценариев нужны данные.
- Тестирование нельзя проводить в производственной среде, нельзя работать с персональными данными.
- Есть вариант копировать все данные из производственной среды в тестовую с применением анонимизиации, но такой вариант может занимать много времени.
- Часто для тестовых сценариев не нужно много данных, но нужны согласованные данные.

Цель — проектирование системы, обеспечивающей перенос взаимосвязанных данных между базами данных, их анонимизацию, генерацию тестовых данных.

Задачи:

1. определение требований к проектируемой системе,
2. анализ и исследование существующих аналогов,
3. проектирование архитектуры системы,
4. разработка алгоритма переноса и генерации взаимосвязанных данных,
5. разработка языка для описания данных,
6. реализация прототипа системы,
7. анализ полученных результатов.

Система должна иметь следующую **функциональность**:

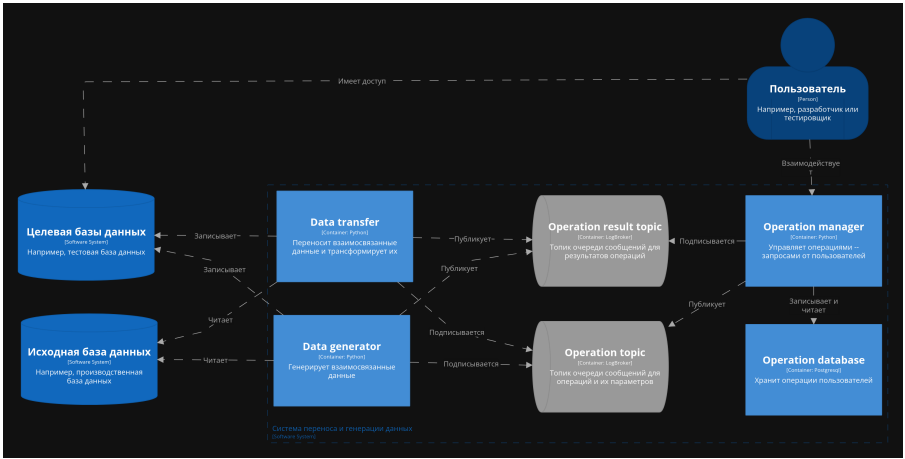
- перенос и анонимизация данных,
- генерация данных.

Архитектура системы должна иметь следующие **свойства**:

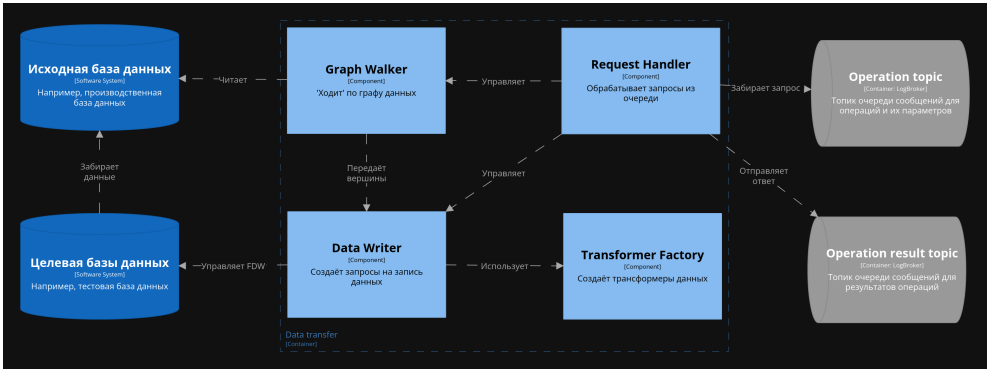
- безопасность,
- производительность,
- надёжность.

Взаимодействие систем

Система переноса и генерации данных



Компоненты Data Transfer

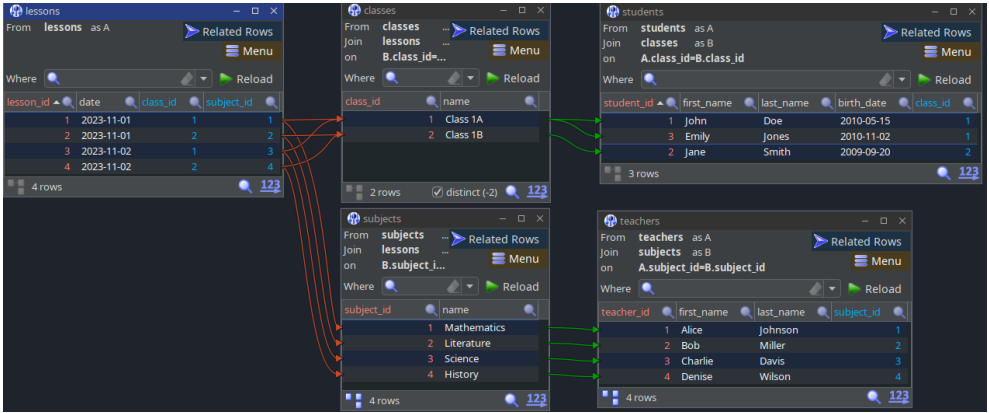


Предпосылки к использованию метаграфов:

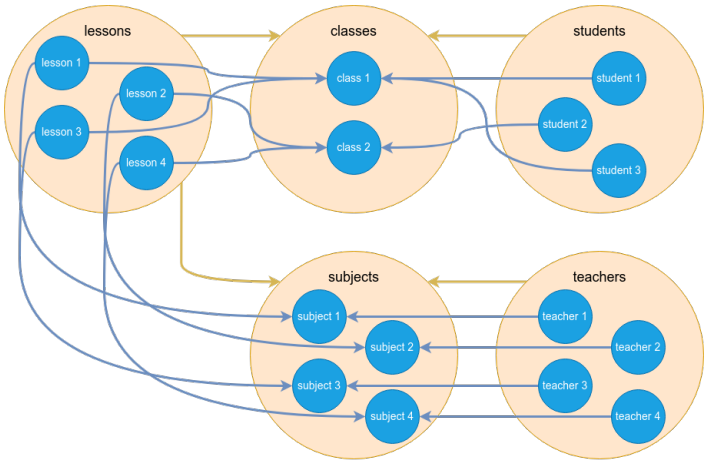
- система данных в БД и их взаимосвязей напоминает граф,
- классические графы не подходят из-за сложной модели объектов в БД.

Пусть $MG = \langle V, MV, E, ME \rangle$ – метаграф, где V – множество вершин, MV – множество метавершин, E – множество рёбер, ME – множество метарёбер.

Пример базы данных



Графическое отображение метаграфа



Алгоритм обхода данных


```
1  Base_Metagraph_Traversal(DB, SV)
2    queue ← ∅
3    visited ← ∅
4    for each v in SV
5      do Enqueue(queue, v)
6    while queue ≠ ∅
7      do cur_v ← Dequeue(queue)
8         Add(visited, cur_v)
9         for each u in Adjacent(DB, cur_v)
10          do if u ∉ visited
11             then Enqueue(queue, u)
12   return visited
```

Алгоритм обхода данных


```
1  Metagraph_Traversal_With_Rules(DB, SV, RME, RE)
2    MG ← INIT(DB)
3    MG.ME ← Apply_Rules(MG.ME, RME)
4    queue ← ∅
5    visited ← ∅
6    for each v in SV
7      do Enqueue(queue, v)
8    while queue ≠ ∅
9      do cur_v ← Dequeue(queue)
10        Add(visited, cur_v)
11        Add(MG.V, cur_v)
12        incident_edges ← Incident(DB, cur_v)
13        new_incident_edges ← Apply_Rules(incident_edges, RE)
14        Extend(MG.E, new_incedent_edges)
15        for each u in Adjacent(MG, cur_v)
16          do if u ∉ visited
17             then Enqueue(queue, u)
18   return visited
```

Примеры описания метаданных для переноса данных

```
1 GRAPH SOURCE classes WHERE class_id=1;
2 NO ENTER teachers;
3 TRANSFORMER random_first_name FOR students.first_name;
4 TRANSFORMER random_last_name FOR students.last_name;
```

```
1 GRAPH SOURCE lessons;
2 EXCLUDE EDGE lessons.class_id classes.class_id;
3 LIMIT VISITS 2 FOR teachers;
4 TRANSFORMER set("Anon") FOR teachers.name;
```

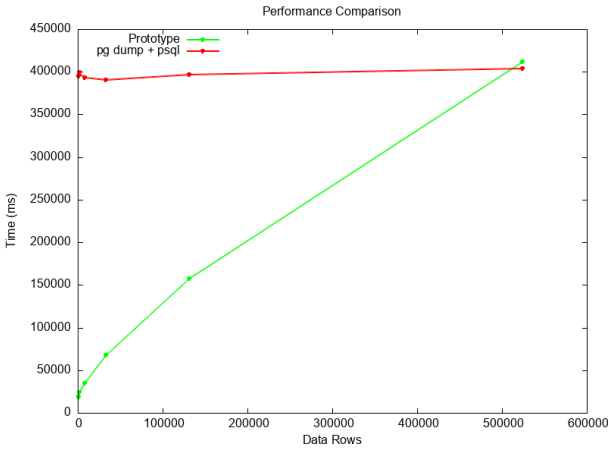
Пример описания метаданных для генерации данных

```
1 GRAPH SOURCE lessons;
2 LIMIT DISTANCE 1 FOR lessons;
3 SET GENERATION VALUES gen_date("2020-01-01", "2025-01-01")
    FOR lessons.data;
4 SET GENERATION VALUES ["Class 1A", "Class 1B", "Class 1C"}
    FOR classes.name;
5 SET GENERATION VALUES gen_random_subject FOR subjects.name;
6 SET GENERATION AMOUNT lessons=10, classes=5, subjects=10;
```

Возможности:

- CLI,
- перенос взаимосвязанных данных,
- работа со схемами.

Тестовая база данных: 4.5 ГБ, 44739072 записей.



Результаты:

- спроектирована безопасная и надёжная архитектура,
- разработан удобный язык описания метаданных, похожий на привычный SQL,
- разработан прототип, который показал высокую скорость работы при переносе небольшого количества взаимосвязанных данных.

Что можно улучшить:

- производительность,
- понимание описания метаданных и алгоритма.

Ссылки на код грамматики и код прототипа (скоро будет).