Университет ИТМО

Факультет программной инженерии и компьютерной техники

Хранилища и базы данных

Лабораторные работы 1, 2, 3

Курсовая работа

Выполнили студенты гр. Р41142: Юнусов А. В.

Нестеров А. А.

Романов О. А.

Преподаватель: Королёва Ю. А.

Санкт-Петербург  
2020

# Задание 1

Создать 4 БД, нормализовать их, где это необходимо. Возможно внесение дополнительных полей. В таблицах, приведённых ниже, в левой колонке представлено описание сведений, содержащихся в БД, которые необходимо разложить по нескольким таблицам. В левой колонке комментарии, поясняющие содержимое полей базы данных.

Должны быть некоторые специализации в одной БД, которые не совпадают с направлением и специализацией в другой БД. В таких ситуациях следует выбрать приоритетный источник данных - для специализаций это Postgres. В итоговую бд следует выбирать значение из источника с большим приоритетом. В случае возникновения других подобных коллизий, приоритет источников вы можете установить самостоятельно

Postgres

|  |  |
| --- | --- |
| Название университета |  |
| Тип стандарта обучения | старый/новый |
| Название дисциплины | Компьютерная графика (2018449043-И) |
| Форма бучения | очная/заочная |
| Место учёбы | факультет |
| Специальность | 09.03.04 – Разработка программно-информационных систем (Академический магистр) |
| Семестр | 7 |
| Лекции | 17 часов |
| Практические занятия | 0 часов |
| Лабораторные | 17 часов |
| Форма контроля | экзамен / зачёт |
| Баллы |  |
| Дата проставления баллов |  |
| ФИО преподавателя |  |
| Идентификатор преподавателя |  |
| ФИО студента |  |
| Идентификатор студента |  |

MySQL

|  |  |
| --- | --- |
| Идентификатор участника |  |
| ФИО участника |  |
| Должность | доцент/бакалавр/магистр |
| Наименование конференции |  |
| Место проведения конференции |  |
| Дата конференции |  |
| Наименование публикации | статья/тезисы |
| Язык издания |  |
| Наименование издания |  |
| Объём издания | 5 листов |
| Место издания | Тверь/Москва |
| Тип издания | вак, ринц …. |
| Соавторы |  |
| Индекс цитирования |  |
| Дата публикации |  |
| Наименование научного проекта |  |
| Участники научного проекта |  |
| Период участия в проекте |  |
| Читательский лист | взял книгу, какую книгу, когда взял, когда вернул |

Oracle

|  |  |
| --- | --- |
| Идентификатор человека |  |
| ФИО человека |  |
| Дата рождения |  |
| Место рождения |  |
| Подразделение | Мегафакультет / Факультет учёбы (работы) |
| Должность | доцент |
| Период работы | 01.09.2000-31.08.2001 |
| Расписание | дисциплина, преподаватель, время, аудитория |
| Группа обучения | 01.09.2000-31.08.2001 1 курс  01.09.2001-31.08.2002 2 курс  01.09.2002-31.08.2003 3 курс |
| Оценка |  |
| Буква |  |
| Дата получения оценки |  |
| Вид обучения | бюджет/контракт |
| Форма обучения | очная/заочная |
| Направление | 09.04.04 – Программная инженерия |
| Специальность | 09.03.04 – Разработка программно-информационных систем (Академический магистр) |
| Квалификация | магистр |
| Учебный год | 2018/2019 |

Mongo DB

|  |  |
| --- | --- |
| Идентификатор человека |  |
| ФИО человека |  |
| Имеет льготы | Y/N |
| Вид обучения | бюджет/контракт |
| Сумма оплаты |  |
| Период посещения | зашёл в общежитие/вышел из общежития |
| Местоположение | пер. Вяземский, наб. р. Карповки ….. |
| Номер комнаты |  |
| Количество комнат в здании |  |
| Комната | на 3 человека |
| В комнате живёт | 2 человека |
| Когда проводили дезинфекцию дата |  |
| Клопы | Y/N |
| Предупреждения | 1, 2, ….. |
| Период с/по | Живёт со 2 курса, то есть  01.09.2001-31.08.2002 2 курс |

## Этап 1. Создание схем.

На основе данных таблиц были разработаны соответствующие схемы в различных базах данных. Для всех баз кроме MongoDB данные схемы были сгенерированы заранее, т.к. сущности в них содержащиеся, имеют ссылки друг на друга, а также различные типы данных, требующие проверки валидности и ограничений.

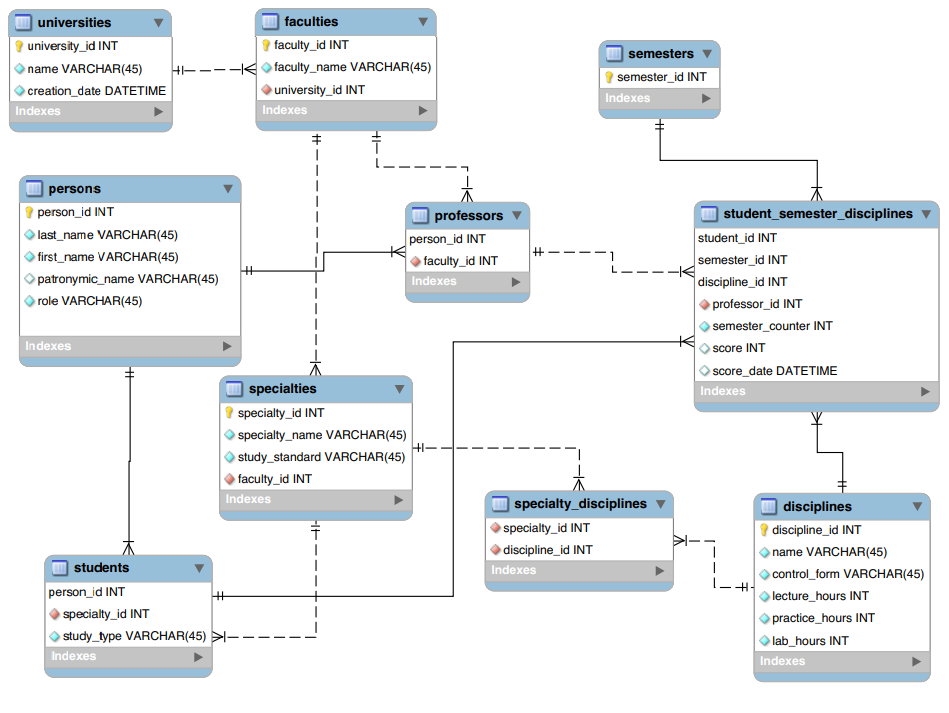
В MongoDB схема не предполагает сложных сущностей с типами данных, требующими дополнительных проверок, поэтому схема для MongoDB будет сгенерирована динамически на основе помещённых в неё данных.

Также была создана общая схема бизнес-сущностей, представляющих отношения между всем сущностями.

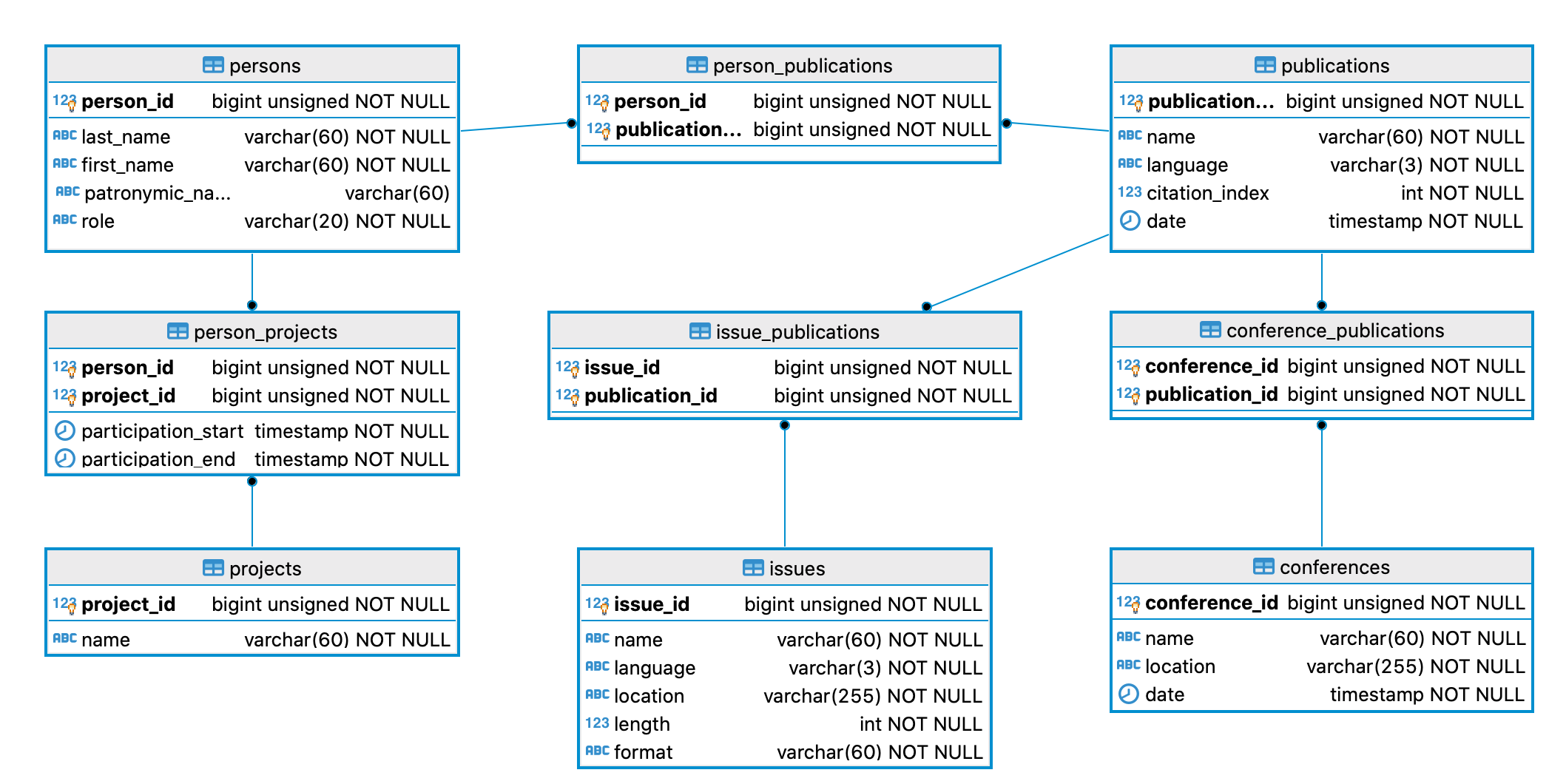
Скрипты создания схем см. в Приложении.

Следует отметить, что для базы Oracle структура размещения данных в таблицу несколько отличается от классической. В базе данных содержится фиксированное количество мета-таблиц, в которых объявляются типы данных, экземпляры и их содержимое.

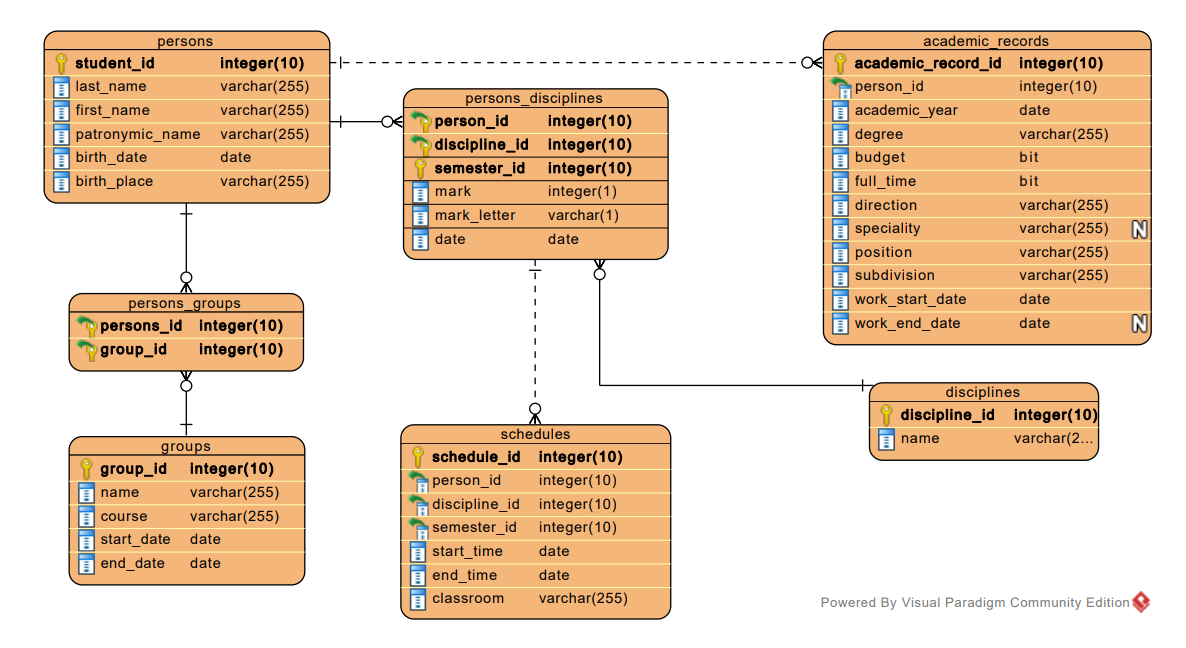
* ITMO\_OBJECT\_TYPES содержит все типы данных
* ITMO\_OBJECTS содержит экземпляры объектов
* ITMO\_ATTRIBUTES содержит данные о полях типов данных
* ITMO\_PARAMS содержит данные экземпляров объектов
* ITMO\_LIST\_VALUES содержит перечислимые типы данных

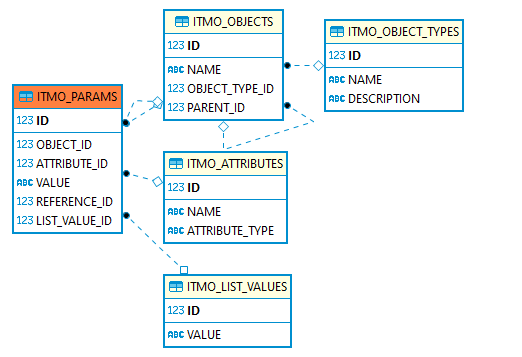
Postgres  


MySQL

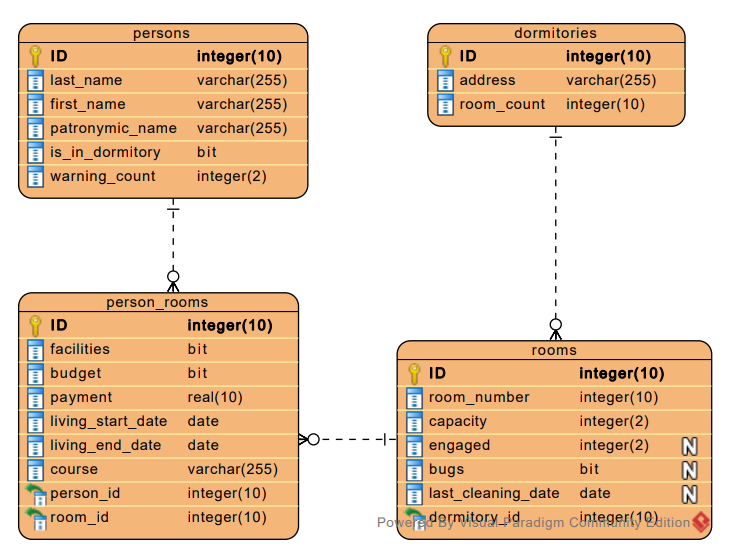


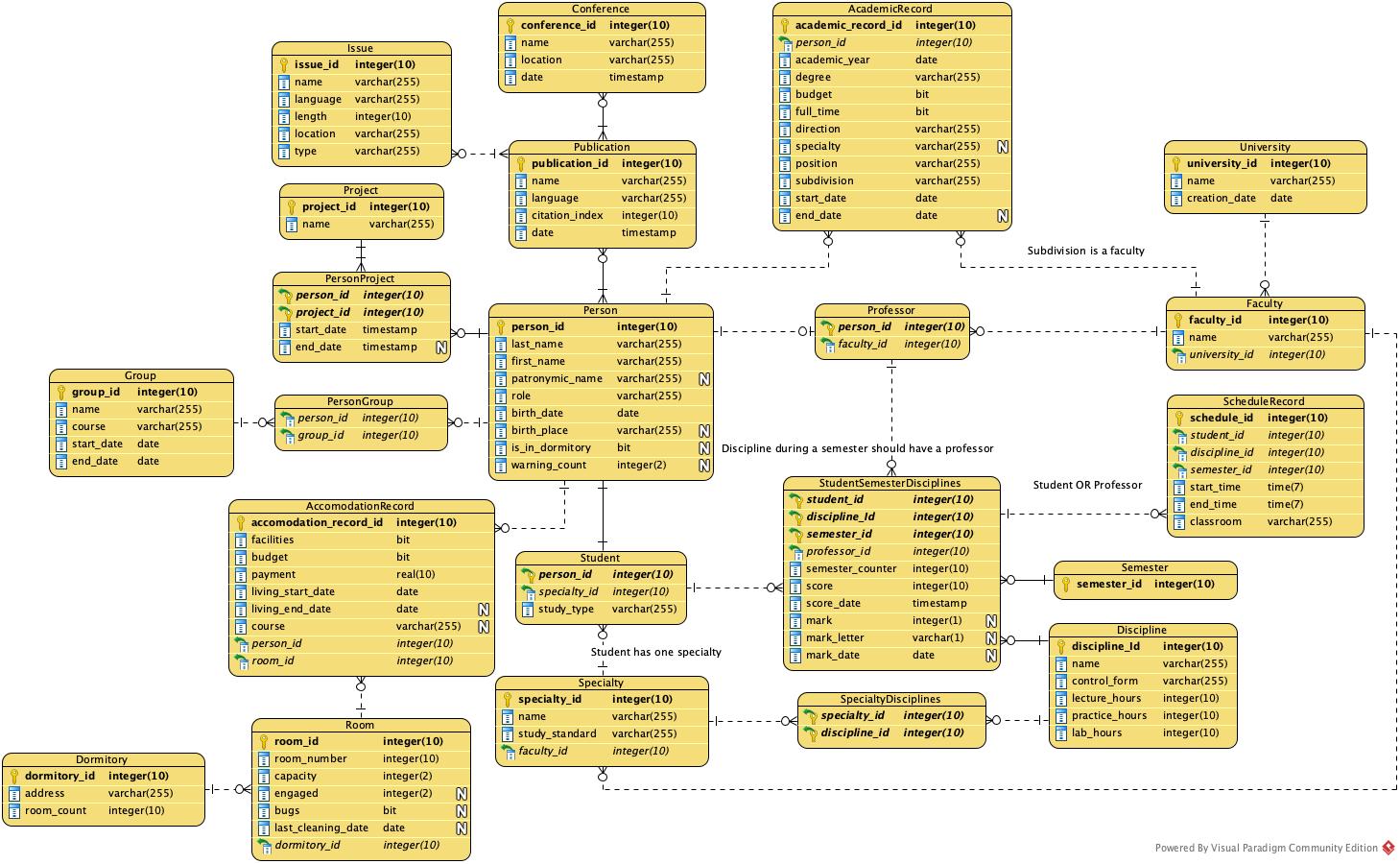
Oracle Entities schema



Oracle DB schema

Mongo



Overall

## Этап 2. Генерация данных

Для заполнения созданных схем данными было разработано приложение-генератор (ссылка на репозиторий: <https://github.com/d32f123/sadb-labs/tree/feature/generator>).

Приложение написано на Java с использованием Spring и JPA.

Так как идея курсового проекта предполагает в дальнейшем маппинг сущностей с нескольких баз в одну, в первую очередь было решено создать общую схему отношений сущностей, опираясь на которую, будут созданы базовые классы сущностей.

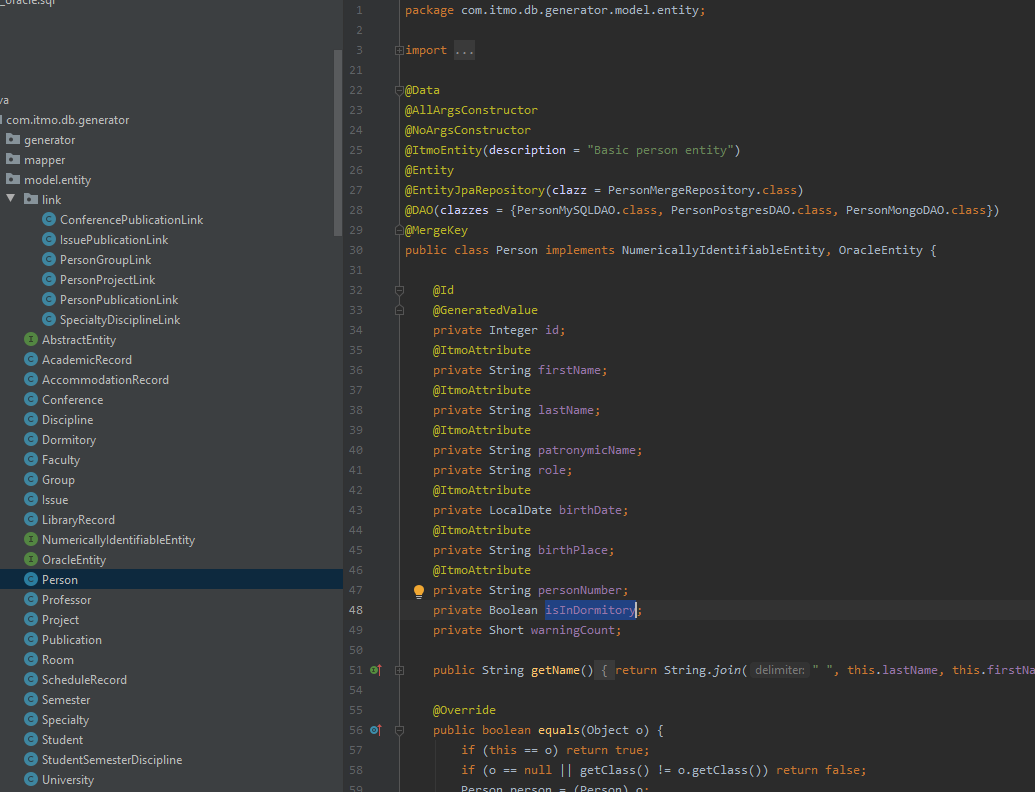
Каждому классу сущности в соответствие был поставлен один или несколько DAO классов, отражающие часть его полей в определённую базу данных.  


Рисунок . Пример базового класса сущности. В строке 28 установлено соответствие между классом сущности и классом DAO.

На базе созданных сущностей были созданы классы-генераторы полей каждой сущности со сложной логикой.

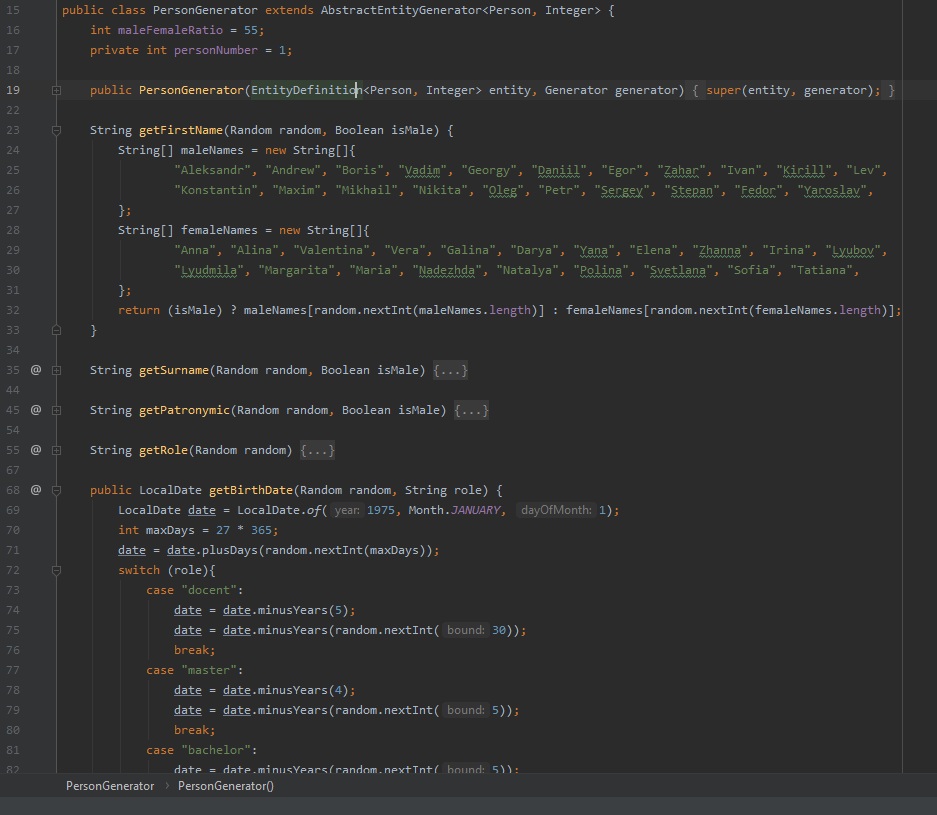


Рисунок . Пример класса-генератора сущности. В строке 16 находится пример метапараметра для сущности – соотношение биологического мужского пола к женскому. В строках 24-31, установлен набор возможных значений поля. В строке 69-81 приведен пример генерации даты рождения человека на основе его роли.

На основе сгенерированных базовых сущностей заполняются их соответственные DAO сущности и с помощью классов-персисторов помещаются в базу. Задача создания сущностей, генерации их содержимого, разбиение их на DAO-сущности и сохранение в соответствующую базу исполняются набором рабочих потоков.

Особую сложность представляет задача генерации связей между сложными сущностями (напр. PersonGroup). Для решения подобных задач был также реализован алгоритма анализа отношений сущностей, который определял порядок генерации данных (напр. сначала генерировался Person и Group, а затем – PersonGroup).

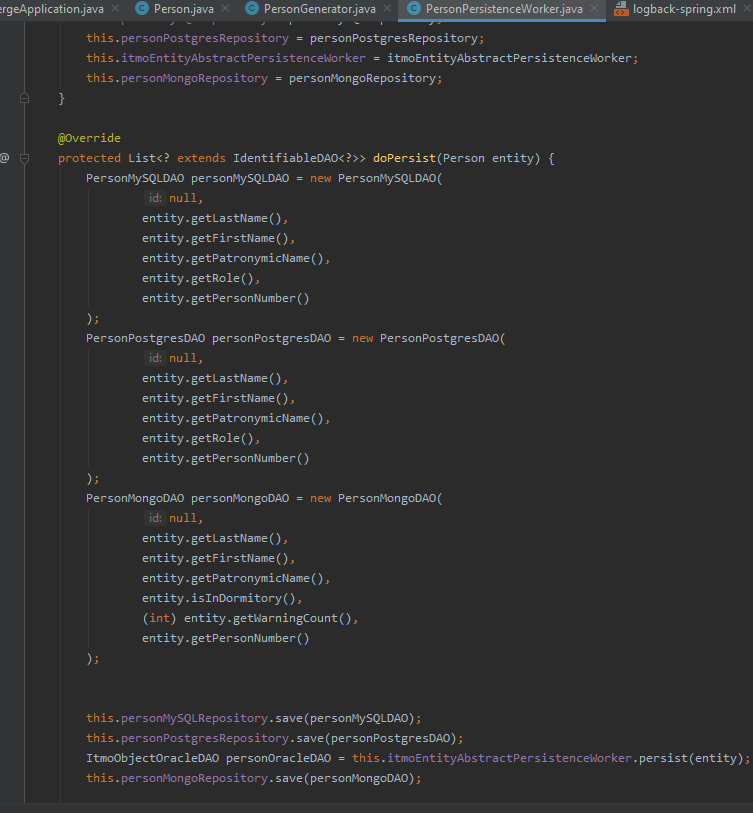


Рисунок . Пример заполнения DAO сущности на основе базовой сущности.

# Задание 2

Из четырёх созданных СУБД необходимо загрузить данные в одну итоговую схему БД (на Oracle).

Для заполнения данными из различных источников в один было разработано приложение-мерджер (ссылка на репозиторий: <https://github.com/d32f123/sadb-labs/tree/feature/merger>)

На основе существующих связей между базовыми сущностями и DAO-сущностями мы обращаемся в соответствующую DAO базу данных в порядке их важности и заполняем часть полей данных сущности.

Особую сложность представляет задача сохранения связей между сложными сущностями (напр. PersonGroup). Для решения подобных задач был повторно использован разработанный алгоритм анализа отношений сущностей.

При генерации данных был использован подход, при котором для каждой сущности создается ряд вспомогательных сущностей (напр. Person, PersonMySQLDAO, PersonPostgresDAO, PersonMongoDAO, PersonGenerator, PersonPersistenceWorker, PersonMySQLRepository, PersonPostgresRepository, PersonMongoRepository). Подобный подход позволяет явно контролировать логику генерации каждого объекта, однако порождает необходимость написания дополнительного кода для каждой сущности.

В случае мерджа данных с нескольких баз, был избран другой подход:  
с помощью аннотаций, а также рефлексии классов, мы получили хорошо масштабируемую логику. Для добавления новой сущности необходимо лишь задать базовый класс, DAO-классы и соответствующий репозитории и с помощью аннотаций создать связь между данными объектами, а логика взаимодействия между ними является общей и генерируется динамически.



Рисунок 4. Фрагмент рефлективной логики. Демонстрирует получение предзагруженных методов из метаданных класса.

В результате была создана общая база данных с сохранением связей между сложными объектами.

# Задание 3.

1. Ознакомиться с выложенной схемой (er-rus for awm.pdf или er-eng for awm.pdf), кто может, дополнить ещё одной таблицей фактов
2. Заполнить схему данными из полученного во втором задании хранилища
3. На основе заполненной схемы создать рабочее место аналитика (AWM), куб, атрибуты и измерения для куба.
4. Получить несколько отчётов

ыыыыыыыыыы

# Курсовой проект

ыыыыыыыыы