**Объектно-орентированные базы данных**

**Лабораторные занятия**

72 часа лабораторных занятий

II семестр — зачет

Темы лабораторных занятий:

1. Концепция объектно-ориентированного программирования. Объектно-ориентированная модель данных.

2. JSON формат. Хранение и обработка объектов с использованием JSON формата.

3. XML формат. XSD схема. Хранение и обработка объектов с использованием XML формата.

4. Объектно-реляционные базы данных.

5. Работа с пользовательскими типами данных в СУБД PostgreSQL.

6. Спецификация JPA.

7. Спецификация JDO.

8. Использование рефлексии для обработки данных объекта.

Результаты лабораторных работ необходимо публиковать в GIT репозитории, используя следующую структуру:

oodb/lab1

oodb/lab2

...

**Лабораторная работа 1**

Тема: Разработка объектно-ориентированной модели данных прикладной задачи.

Цель: Научиться разрабатывать объектно-ориентированную модель данных прикладной задачи.

Задание: Выбрать предметную область. Проанализировать состав информации, подлежащий хранению и обработке. Построить классы на языке Java, соответствующие выявленным сущностям.

Требования: необходимо разработать не менее 4 сущностных классов. Хотя-бы один из классов должен содержать атрибут типа Collection<...>. Классы должны быть связанными (т. е.каждый класс должен иметь атрибут типа другой сущностный класс, или выступать в качестве типа некоторого атрибута другого сущностного класса). Классы должны содержать методы, реализующие внутреннюю логику предметной области.

**Лабораторная работа 2**

Тема: JSON формат. Хранение и обработка объектов с использованием JSON формата.

Цель: изучить JSON формат, изучить способы работы с JSON средствами языка Java.

Задание: разработать приложение сохраняющее объекты (структура лабораторной работы 1) в файл, считывающее объекты из файла в формате JSON, выводящее список сохраненных объектов в отсортированном виде по некоторому критерию. Поиск объекта по некоторому критерию из ранее сохраненного массива.

**Лабораторная работа 3**

Тема: XML формат. XSD схема. Хранение и обработка объектов с использованием XML формата.

Цель: Изучить XML формат, научиться работать с XSD схемами, изучить способы работы с XML средствами языка Java.

Задание: разработать приложение сохраняющее объекты (структура лабораторной работы 1) в файл, считывающее объекты из файла в формате XML, выводящее список сохраненных объектов в отсортированном виде по некоторому критерию. Поиск объекта по некоторому критерию из ранее сохраненного массива.

**Лабораторная работа 4**

Тема: JSON формат в СУБД PostgreSQL. Хранение и обработка объектов с использованием JSON формата.

Цель: изучить способы работы с типом данных JSON и JSONB в СУБД PostgreSQL.

Задание: разработать приложение сохраняющее объекты (структура лабораторной работы 1) в СУБД PostgreSQL, считывающее объекты из поля таблицы в формате JSON, выводящее список сохраненных объектов в отсортированном виде по некоторому критерию. Поиск объекта по некоторому критерию из ранее сохраненного массива.

Для выполнения работы создать базу данных, таблицу с 2 полями типа json и jsonb (для удобства добавить поле id типа int как первичный ключ).

Все операции добавления, чтения, изменения данных в СУБД сопровождать измерением времени выполнения. Сравнить время работы с json и jsonb.

**Лабораторная работа 5**

Тема: пользовательские типы данных PostgreSQL

1) Структуру объектов базы данных перенести в структуру пользовательских типов данных

PostgreSQL (с использованием команды создания типа CREATE TYPE наименование\_типа AS …).

2) Создать приложение Java, наполняющее созданную БД информацией, выполняещее операции обновления данных, выборки данных.

**Лабораторная работа 6**

Тема: Анализ объектной модели данных.

Задание: Просканировать пакет с объектной моделью данных, сформировав сетевую структуру (узлы — классы, ребра — связи с указанием типа). Сформировать UML диаграмму, отражающую полученную структуру.

Для корректного формирования структуры использовать следующие аннотации:

@Entity — аннотация, проставляемая над классом, определяющая его как класс-сущность.

@OneToOne - аннотация, проставляемая над членом класса, определяющая связь «Один к одному».

@ManyToOne - аннотация, проставляемая над членом класса, определяющая связь «Многие к одному».

@OneToMany — аннотация, проставляемая над членом класса, определяющая связь «Один ко многим», удобно использовать для коллекций.

@Column - аннотация, проставляемая над членом класса, определяющая атрибут сущности.

**Лабораторная работа 7**

Тема: Принципы построения ORM

Задание: Сформировать реляционную базу данных, соответствующую объектной модели, исходя из следующих правил:

1. Сущность представлена классом, аннотированным как @Entity

2. Сущность моделируется таблицей, именуемой как lower(ClassName)

3. Члены класса, аннотированные как @Column, соответствуют полям таблицы, названным так-же как члены класса (отдельно оговаривается случай @ManyToOne и @OneToOne). Названия полей таблиц можно задавать атрибутом name аннотации @Column

4. Класс имеет член, соответствующий первичному ключу, и аннотированный как @Id (рекомендуется использовать в качестве типа такого члена — Long), именованный как id

5. Классы, являющиеся потомками в иерархии наследования, отображаются на таблицы, содержащие поля предков этих классов

6. Члены класса, аннотированные как @ManyToOne или @OneToOne соответствуют полю таблицы, названному по имени члена класса с окончанием «\_id». Тип этого поля должен соответствовать типу первичного ключа связанной таблицы

7. Члены класса с аннотациями @OneToMany или @ManyToMany, имеющие тип List<...>, описывают множественные связи к сущности. Соответствие с реляционной моделью необходимо будет реализовать с помощью таблицы посредника, содержащей два поля со значениями первичных ключей двух связанных сущностей. Таблицу-посредник именовать как «lower(ClassName)\_lower(ListClassName)». Поля этой таблицы «lower(ClassName)\_id» и «lower(fieldName)\_id»

Написать программу, проверяющую корректность реляционной БД (проверить наличие таблиц, соответствующих классам-сущностям; проверить наличие полей, соответствующих членам классов, без проверки типов)