**Объектно-орентированные базы данных**

**Лабораторные занятия**

72 часа лабораторных занятий

II семестр — зачет

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).

**Лабораторная работа 8**

Тема: Управляющие конструкции ORM

Задание:

1. реализовать интерфейс EntityManager (см. ), управляющий сущностями базы данных.

Основные задачи класса, реализующего этот интерфейс — поддержка связи между сущностными классами и их представлением в реляционной БД. Класс должен реализовать методы сохранения, удаления, обновления данных в БД посредством манипулирования с сущностными классами.

Реализация EntityManager должна иметь сформированное подключение к БД (переменная типа Connection) через которое обращаться к БД.

Реализация EntityManager должна иметь список (List<Object>) сущностных классов, связанных с БД (т. е. таких, которые были сохранены, обновлены, прочитаны через EntityManager).

2. создать класс EntityManagerFactory, основной задачей которого будет создание экземпляра реализации EntityManager. Помимо этого класс EntityManagerFactory должен создать подключение к БД (или пул подключений), просканировать модель и БД на соответствие друг другу.

3. Написать программу, работающую с БД через EntityManager: создание экземпляров классов-сущностей, сохранение их в БД, изменение, получение экземпляра сущности из БД.

**Лабораторная работа 9**

**Тема**: Принципы проектирования объектно-ориентированной БД с использованием спецификации JPA.

**Задание 1**: По имеющейся предметной области создать объектную модель данных с использованием требований спецификации JPA. Создать программу, генерирующую структуру реляционной БД в СУБД PostgreSQL.

**Лабораторная работа 10**

**Тема**: Реализация операций CRUD в рамках объектно-ориентированной модели данных с использованием технологий ORM.

**Задание 1**: Создать программу, реализующую операции CRUD в рамках объектно-ориентированной модели данных с использованием технологий ORM.

**Лабораторная работа 11**

**Тема**: Работа с первичным ключом.

**Задание 1**: Разработать объектную модель данных, в сущностях которых определен фиктивный целочисленный первичный ключ с автоматической его генерацией. Задействовать 3 стратегии генерации.

**Лабораторная работа 12**

**Тема**: Работа с механизмом миграции.

**Задание 1**: Создать приложение, генерирующее базу данных по объектной модели. Подключить механизм миграции данных (библиотека flywaydb). Создать SQL скрипт модифицирующий базу данных (по структуре или по данным) для миграции. Запустить приложение, убедиться в успешном выполнении миграции.

**Лабораторная работа 13**

**Тема**: Отображение иерархических структур (наследование) на реляционные БД

**Задание 1**: Создать объектную модель данных в которой должна быть иерархическая 3-х уровневая структура, например:

public abstract class AbstractEntry {

public abstract Long getId();

public abstract void setId(Long id);

}

@ … // Здесь указывается аннотация, соответствующая заданию

public class Person extends AbstractEntry {

@Id

private Long id;

...

}

@Entity

public class Staff extends Person {

…

}

@Entity

public class Client extends Person {

…

}

В модели должны быть минимум 2 класса верхнего уровня (в примере это Staff и Client).

**Задание 2:**

Сгенерировать структуру реляционной БД, основанную на стратегии наследования MappedSuperClass, разработать приложение, сохраняющее данные из верхних по иерархии классов, убедиться в корректном сохранении данных в СУБД.

**Задание 3:**

Сгенерировать структуру реляционной БД, основанную на стратегии наследования Inheritance(strategy = *JOINED*), разработать приложение, сохраняющее данные из верхних по иерархии классов, убедиться в корректном сохранении данных в СУБД.

**Задание 4:**

Сгенерировать структуру реляционной БД, основанную на стратегии наследования Inheritance(strategy = *SINGLE\_TABLE*), разработать приложение, сохраняющее данные из верхних по иерархии классов, убедиться в корректном сохранении данных в СУБД, исследовать содержимое поля dtype таблицы.

**Задание 5:**

Сгенерировать структуру реляционной БД, основанную на стратегии наследования Inheritance(strategy = *TABLE\_PER\_CLASS*), разработать приложение, сохраняющее данные из верхних по иерархии классов, сохраняющее данные из классов 2-го уровня иерархии, убедиться в корректном сохранении данных в СУБД.

**Вопрос**: Какая стратегия соответствует более высокому уровню нормализации реляционной БД.