

Задание 2. Приложение на основе Hibernate, создающее таблицы в базе данных и работающее с данными в этих таблицах

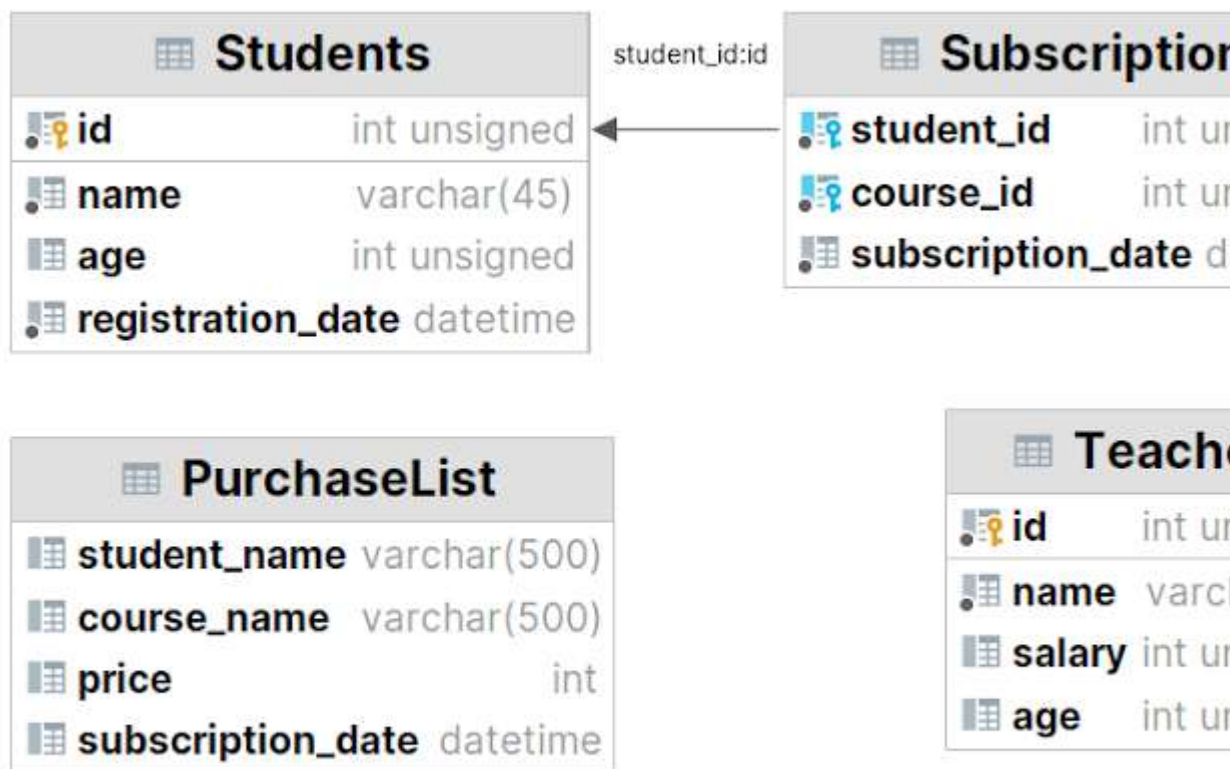
Цели задания

Научиться:

- подключать к проекту библиотеку Hibernate;
- создавать классы @Entity со связями;
- создавать таблицы в базе данных при помощи Hibernate и заполнять их данными.

Что нужно сделать





- Создайте новый проект в папке SQLAndHibernate из репозитория java_basics. В папке config есть пример XML-файла конфигурации Hibernate.
- Создайте отдельную пустую базу данных MySQL, к которой будете подключаться в этом проекте и с которой будете работать.
- Подключите библиотеку Hibernate к вашему проекту так, как показано в видео.
- Создайте классы @Entity для всех таблиц и пропишите связи между ними в соответствии со схемой:



- Запустите приложение, чтобы в базе данных автоматически появились созданные вами таблицы.
- Залейте в эти таблицы готовый дамп с данными.
- Создайте класс `LinkedPurchaseList` для таблицы со следующей структурой:
 - `student_id`
 - `course_id`
- Конвертируйте данные таблицы `PurchaseList` в данные для таблицы `LinkedPurchaseList`, в которой уже хранятся идентификаторы студентов и курсов: пары значений `student_id` и `course_id`. Напишите соответствующий код, запустите его и убедитесь, что он выполнен верно.

В таблице `LinkedPurchaseList` не предполагается отдельной колонки с привычным идентификатором записи. В роли идентификатора здесь выступает пара `student_id` и `course_id`.

Это значит, что пара значений `student_id` и `course_id` уникальна для каждой записи. Такая пара позволит получить всех студентов курса по `course_id` или найти все курсы одного студента по `student_id`, а также связать эти `id` с данными таблиц `Students` и `Courses`.

	 <code>student_id</code> 	 <code>course_id</code> 
1	1	22
2	2	3
3	2	5
4	1	7
5	3	22

Если мы попробуем добавить ещё одну запись `student_id=3` и `course_id=22`, то база данных не позволит сделать это при составном ключе, так как такая пара уже есть.

Такая связка `student_id` и `course_id` уникальна, она называется составной ключ (composite key).

Один из вариантов создания составного ключа — написать отдельный класс, в котором поля будут содержать значения полей, входящих в составной ключ, и уже этот класс будет использоваться как основной ключ Entity.

Рассмотрим наш случай и создадим класс для составного ключа `Subscription`:

```
public class SubscriptionKey implements Serializable {
    @Column(name = "student_id")
    private int studentId;

    @Column(name = "course_id")
    private int courseId;

    //setters, getters, equals(), hashCode()
}
```

Класс-ключ в @Entity используется следующим образом:

```
@Entity
@Table(name = "Subscriptions")
public class Subscription {
    @EmbeddedId
    private SubscriptionKey id;

    @Column(name = "student_id", insertable = false,
updatable = false)
    private int studentId;

    @Column(name = "course_id", insertable = false,
updatable = false)
    private int courseId;

    //other fields, setter, getters
}
```

Аннотация @EmbeddedId говорит, что этот параметр является составным ключом.

Если поля ключа использовать и в основном классе @Entity, то необходимо запретить использование полей для вставки и обновления данных дополнительными параметрами insertable и updatable в аннотации @Column. Для вставки значений ключа используйте поля объекта SubscriptionKey id.

Чтобы получить объект из базы данных по составному ключу, необходимо в метод get() объекта Session передавать класс получаемого объекта и экземпляр составного ключа, например:

```
Subscription subscription =
session.get(Subscription.class, new
SubscriptionKey(studentId, courseId));
```

Аналогично и для создания новой записи: создаём объект класса SubscriptionKey и, используя сеттер, устанавливаем значение в @Entity-класс.

Дополнительные примеры — в статье [«Первичные ключи в Hibernate»](#).